AD-785 260

3-D STRESS WAVE CODE FOR THE ILLIAC IV

Gerald A. Frazier, et al

Systems, Science and Software

Prepared for:

Defense Advanced Research Projects Agency

26 July 1974

DESTROY THIS REPORT WHEN IT IS NO LONGER NEEDED.
DO NOT RETURN TO SENDER.

Field Command Printing Division

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1 REPORT NUMBER DNA 3331F	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER AD-785260
4 TITLE (and Subtitle) 3-D STRESS WAVE CODE FOR THE ILLIAC IV		5. TYPE OF REPORT & PERIOD COVERED Final Report 6. PERFORMING ORG. REPORT NUMBER R-2103
7 AUTHOR(s) Gerald A. Frazier Christian M. Petersen		8. CONTRACT OR GRANT NUMBER(s) DNA 001-72-C-0154
9 PERFORMING ORGANIZATION NAME AND ADDRESS Systems, Science and Software, Inc. P.O. Box 1620 La Jolla, California 92037		10 PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS DARPA Order No. 2102 Program Code No. 3F10 Work Unit No. 01
11 CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency Arlington, Virginia 22209		12 REPORT DATE 26 July 1974 13. NUMBER OF PAGES
14 MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Director Defense Nuclear Agency Washington, D.C. 20305		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16 DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17 DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18 SUPPLEMENTARY NOTES

This work sponsored by the Defense Advanced Research Projects Agency under DARPA Order No. 2102.

19 KEY WORDS (Continue on reverse side if necessary and identify by block number)

ILLIAC IV
Finite Element
Seismic
Stress Waves
3-D Geometry
Curvilinear Coordinates

20 ABSTRACT (Continue on reverse side if necessary and identify by block number)

A novel finite element scheme has been developed for simulating stress waves using 1-D, 2-D, and 3-D orthogonal curvilinear coordinate systems. Finite deformations and nonlinear material behavior are accommodated using a Lagrangian formulation. In addition, the explicit time stepping scheme is efficient; computing rates of 0.4 and 1.2 m-sec per element per numerical time step have been achieved

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. ABSTRACT (Continued)

in 2-D and 3-D test calculations on the ILLIAC IV. About 50 percent slower computing rates were obtained in test calculations performed in 2-D and 3-D spherical coordinate systems.

PREFACE

This formal technical report entitled "3-D Stress Wave Code for the ILLIAC IV," is submitted by Systems, Science and Software (S^3) to the Advanced Research Projects Agency (ARPA) and to the Defense Nuclear Agency (DNA).

The report presents the results of a continued effort to develop a versatile numerical scheme for simulating 3-D stress waves on the ILLIAC IV computer system.

This work was supported by the Advanced Research Projects Agency and was monitored by the Defense Nuclear Agency under Contract No. DNA 001-72-C-0154. Colonel David C. Russell has been the ARPA Program Manager and Lt. Colonel F. J. Leech has been the DNA Project Scientist.

Dr. Gerald A. Frazier has been the S^3 Project Manager. Much help and advice has been obtained from the User Support Group of the IAC. The authors are also grateful for many valuable communications with other users of the ILLIAC IV and the ARPANET, particularly those with Terry Layman of the CAC.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (Continued)

LIST OF TABLES

I. INTRODUCTION

Over the past twenty months, we have formulated, developed, and implemented two numerical computer codes for processing stress wave calculations on the ILLIAC IV computer. Our primary goal has been to develop a capability for performing 3-D wave calculations with a spatial resolution that is comparable to conventional 2-D calculations.

Such a computing capability would most certainly prove to be a valuable asset in numerous ground motion studies. The Lagrangian stress wave code that is being developed on the ILLIAC, referred to as SWIS (Stress Waves In Solids), is expected to have important applications for simulating seismic phenomena such as:

- Explosions in prestressed and geologically complex formations.
- Spontaneous earthquake ruptures and near-field ground motions.
- Stress waves passing through laterally varying earth models.
- Stress waves impinging on buried and surface structures.

This 3-D simulation capability is expected to play an important role in the Advanced Research Projects Agency's (ARPA) program to discriminate earthquakes from explosions and the Defense Nuclear Agency's (DNA) investigations on the vulnerability of buried structures to incoming stress waves.

In order to simulate the seismic phenomena itemized above in three spatial dimensions, the capability must exist for handling very large grids. It is our opinion that, with the advent of super computers such as the ILLIAC IV and highly sophisticated numerical computing algorithms, detailed

3-D wave calculations are now feasible. Based on predicted computing speeds for the ILLIAC, we have estimated that it is theoretically feasible to process wave calculations at the rate of 0.10 m-sec of computer time per numerical time step per 3-D element. This amounts to 10 secs of computer time per numerical time step for a 3-D grid containing 100-thousand elements (e.g., a 50 by 50 by 40 grid). This computing rate is nearly comparable to that achieved in 2-D wave calculations for an equivalent 50 by 50 grid using a conventional serial computer, e.g., a UNIVAC 1108.

Numerically processing wave calculations in 3-D grids with more than 100-thousand elements involves an enormous number of calculations. More than one billion floating point multiply and add operations can arise from a single computer simulation. Clearly, considerations of computing efficiency become extremely important in designing and implementing such a computing scheme. Also, the usefulness of such a code, once it is finalized, will depend on its flexibility for handling a range of geometric configurations, boundary conditions, material types, etc.

In an effort to arrive at an optimum computing scheme, we have carefully reviewed existing numerical computing techniques, both finite element and finite difference, in order to combine strong points from each method into a single algorithm in a form that is suited for parallel processing on the ILLIAC. In so doing we have conceived a hybrid scheme that employs the finite element method for spatial discretization (employing spatial interpolation functions and a virtual work expression) and follows a computing sequence that resembles Lagrangian finite difference schemes; the constitutive properties of the material appear in an isolated module of the code so that the nonlinear flow rule can easily be altered. In addition, the SWIS code, which employs the

hybrid algorithm, contains some advanced features that will enhance its usefulness in particular applications:

- The code operates, with no loss of efficiency, in one-, two-, or three-spatial dimensions.
- The code contains a flexible grid generator which can be superseded in local portions of the grid.
- The calculations are performed in orthogonal curvilinear coordinates.
- The code has a provision for effectively suppressing wave reflections at grid boundaries.

A detailed description of the SWIS computing scheme is presented in Section II of this report.

During the early stages of code development, the ILLIAC IV was not operational. The first successful program execution on the array took place in March, 1973. Because no I/O facilities were available at this time, computed results had to be extracted from core dumps. A linearized version of SWIS (described in Section 3.2 of this report) first became operational on the array in July, 1973. A number of plane-wave calculations were performed at this time to test various features of the linear SWIS code, e.g., artificial damping and transmitting boundary conditions. One calculation was performed which involved 10,000 time steps in an effort to test the stability of the current ILLIAC configuration and to obtain estimates of computing rates.

Following the initial success with the linearized version of SWIS, ILLIAC program development for the more

involved nonlinear SWIS code was begun in September. As the result of improvements in the ILLIAC computing system and our earlier experiences in implementation on the system, this code development work progressed rapidly. The nonlinear SWIS became operational for 1-D geometries in late November, and, during the month of December, both 2-D and 3-D wave calculations were performed on the ILLIAC. Based on the central system clock times, we estimate that the general SWIS code is processing wave calculations at the rate of 1.2 m-sec of computer time per numerical time step per 3-D element. The entire code is programmed in GLYPNIR, and, although nearly all of the calculations are carried out in parallel, no effort has been made to optimize the resulting machine code. Also, we note that the ILLIAC should ultimately process calculations much faster than in its present configuration; perhaps a factor of three or four will be realized from software (overlapping machine instructions) and hardware improvements that are being considered. Thus, we are anticipating somewhat faster computing rates in the future. Our goal has been set at 0.10 m-sec of ILLIAC time per numerical time step per 3-D element.

II. NUMERICAL ALGORITHM

2.1 INTRODUCTION

The numerical algorithm that has been designed for the SWIS code contains features of both finite element and finite difference methods. In many respects, it is like a finite element method in that the continuum is discretized using spatial interpolation functions and a virtual work principle, but the computing sequence is modeled after Lagrangian finite difference shock codes. Figure 2.1 illustrates how three distinct steps in a Lagrangian finite difference code, one of which involves the constitutive properties, are combined into one step in the conventional finite element method through the use of a stiffness matrix. The SWIS code does not develop the finite element stiffness matrix but rather directly computes strain rate, stress, and restoring forces.

2.2 PROBLEM INITIALIZATION

The following quantities are needed to pose the stress wave calculation:

1. Coordinate System Designation

   a. Number of spatial dimensions to appear in the numerical grid.

   b. Orthogonal curvilinear coordinate system to be employed in the calculations. Transformation metrics are provided internally for operating in Cartesian, spherical, and cylindrical coordinate systems.

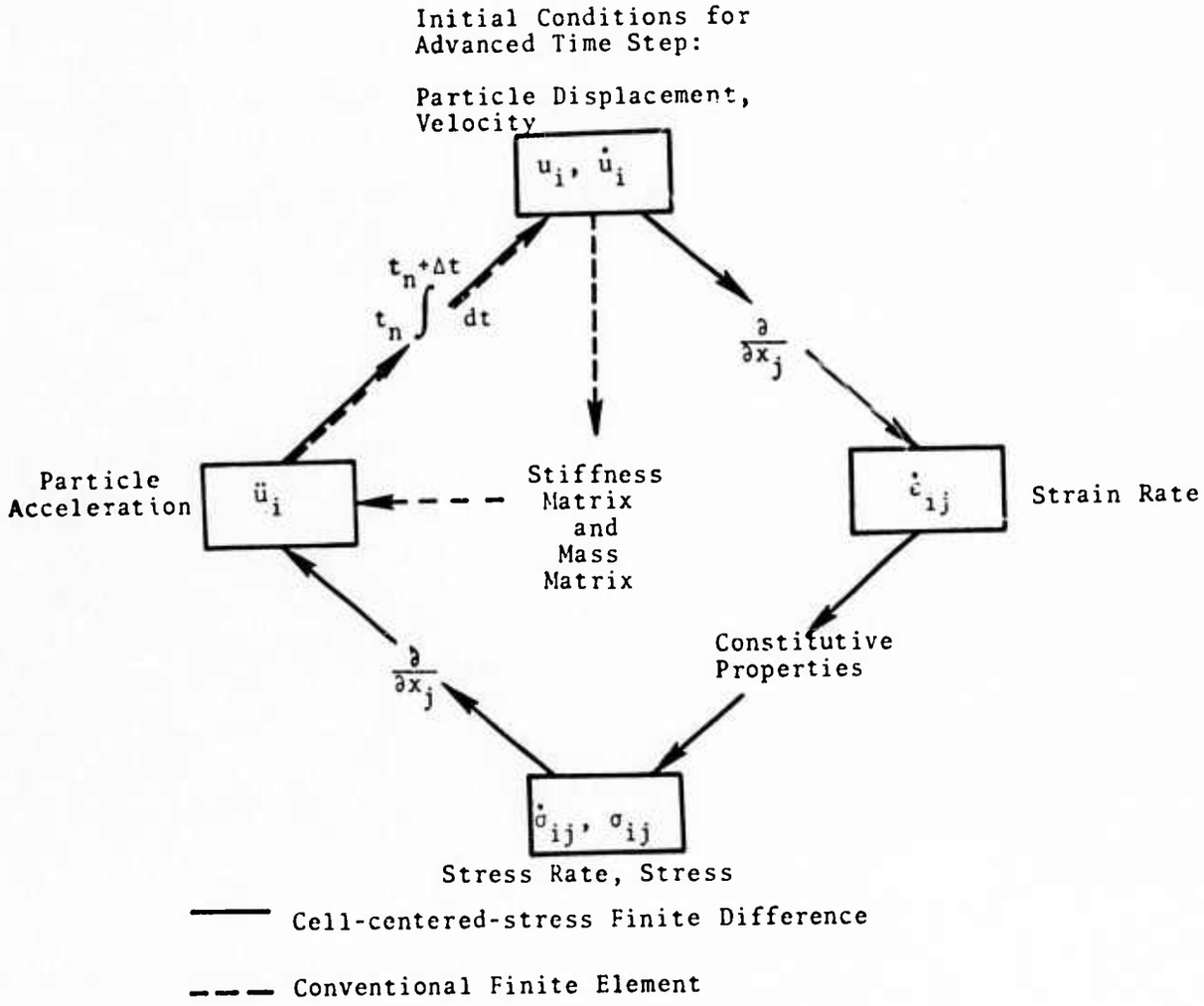


Fig. 2.1--Schematic of the computational sequence for propagating stress waves using conventional finite element and finite difference methods.

2. Grid Configuration

Most grids can be produced using a flexible code-contained grid generator; however, a provision is made for superseding the grid generator in local regions. The grid configuration is described by:

a. Spatial location of the node points, and

b. Node map to associate nodes with elements.

3. Boundary Conditions and Applied Forces

No distinction is made between internal nodes and boundary nodes in that each directional component of each node point is assigned one of the following three constraint conditions:

a. Unconstrained with applied body force or surface traction to form an array of nodal forces.

b. Constrained with nodal displacement components constrained to follow a specified time history (moving or stationary).

c. Transparent with a boundary disguised to reflect almost no incident wave energy.

4. Material Properties

Each element is associated with a material described by

a. Density.

b. Constitutive properties, i.e., properties for developing stress rate as a function of strain rate and stress.

c. Dimensionless coefficient for regulating the damping of spurious high frequency numerical oscillations.

5. Time Stepping Data

   a. Starting time and final time.

   b. Time step Δt.

6. Starting Conditions

   a. Velocity and displacement with respect to some reference frame.

   b. Stress at the centroid of each element.

7. Presentation of Results

   a. Element and node numbers for which results are to be printed at designated time intervals.

   b. Printer plots for displaying results at designated time intervals.

   c. Time histories of individual node points.

   d. Plot files producing graphical displays of the computed results.

Default conditions and data generation schemes are used where possible to minimize the quantity of data that is needed to describe a problem.

2.3 SPATIAL DISCRETIZATION

A spatial region is discretized by subdividing its volume V into a total of E elements, illustrated in Fig. 2.2. The displacement field, $u_i(\underset{\sim}{x},t)$, throughout V, is interpolated from spatially sampled displacements $u_i(\underset{\sim}{X}_n,t)$ where $\underset{\sim}{X}_n$, $n = 1, 2, \ldots N$, are isolated node points which are positioned at juncture points along element boundaries; N is the total number of node points in the numerical grid. The spatial interpolation is achieved using piecewise smooth interpolation functions $p_n(\underset{\sim}{x})$ so that

$$u_i(\underset{\sim}{x},t) = \sum_{n=1}^{N} p_n(\underset{\sim}{x})\ u_i(\underset{\sim}{X}_n,t) \tag{2.1}$$

in which

$$p_n(\underset{\sim}{X}_m) = \delta_{nm}$$

Spatial derivatives of the displacement field are then expressed in terms of nodal displacement by the appropriate differentiation of Eq. (2.1).

$$\frac{\partial u_i}{\partial x_j}(\underset{\sim}{x},\ t) = \sum_{n=1}^{N} \frac{\partial p_n}{\partial x_j}(\underset{\sim}{x})\ u_i(\underset{\sim}{X}_n,t) \tag{2.2}$$

2.3.1 Orthogonal Curvilinear Coordinates

Curvilinear coordinates are often better suited for particular applications than Cartesian; notably, cylindrical and spherical coordinates are well suited for explosion calculations. The capability to operate in spherical (or spheroidal) coordinates also has value for applications in global seismology. In order to accomodate applications

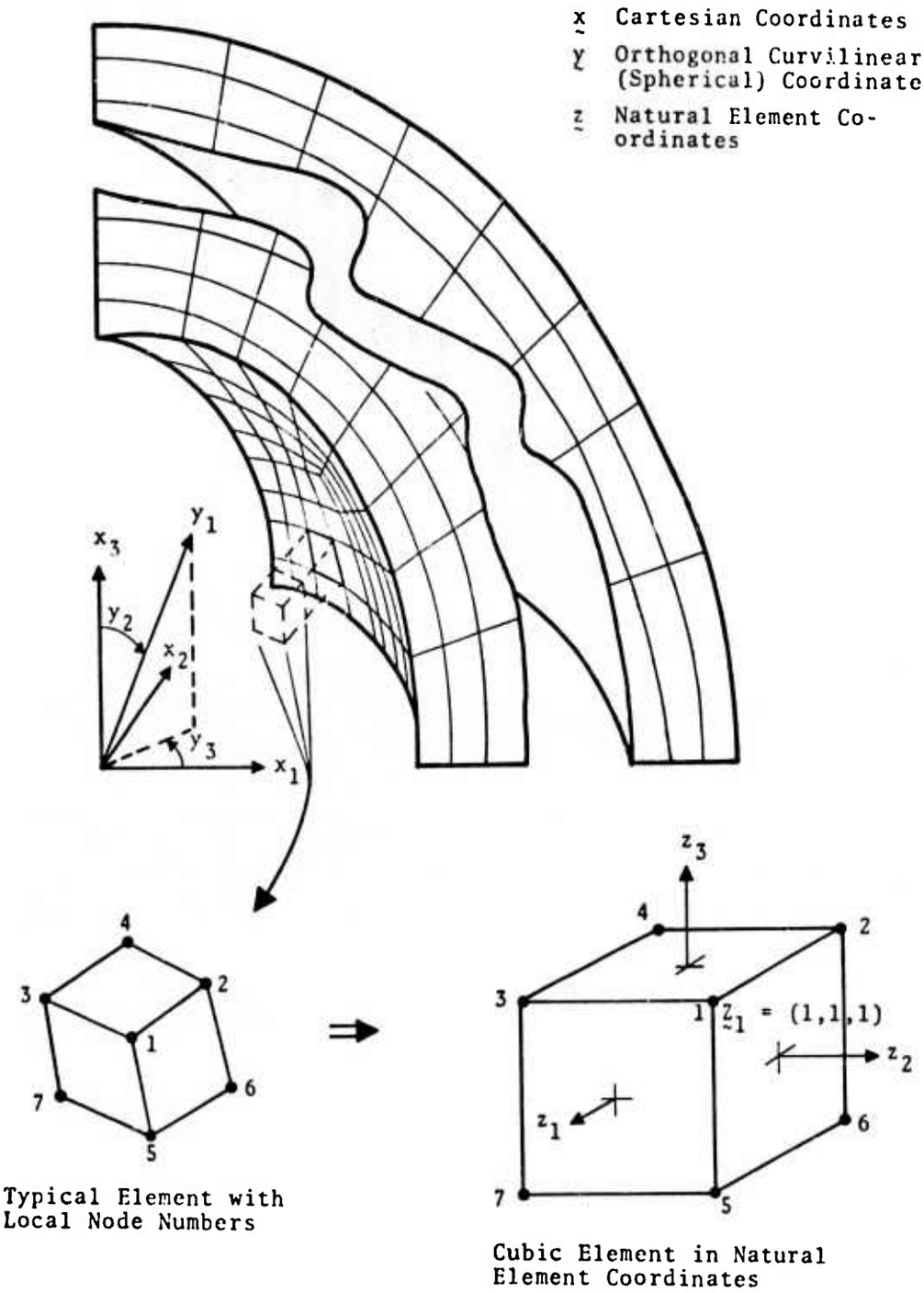


Fig. 2.2--Three-dimensional grid illustrating coordinate systems.

which can benefit from the use of a particular coordinate system, the SWIS code has been developed to operate in general orthogonal curvilinear coordinates designated $\underset{\sim}{y}$ (we reserve the notation $\underset{\sim}{x}$ for Cartesian systems). A brief development of how this is accomplished is presented below.

Consider the point P with Cartesian coordinates $\underset{\sim}{x}$ and curvilinear coordinates $\underset{\sim}{y}$. We define a local Cartesian system, $\underset{\sim}{x}'$, with its origin centered at P. Each component of x_i' is measured in the direction tangent to the respective curvilinear component y_i at point P, illustrated in Fig. 2.3 The metric coefficients for the orthogonal curvilinear system $\underset{\sim}{y}$ are given by

$$h_i = \frac{\partial x_i'}{\partial y_{(i)}} \tag{2.3}$$

where the brackets enclosing the subscript indicate that the summation convention does not apply to that particular subscript. Table I contains metric coefficients and their curvilinear spatial derivatives for some of the more common orthogonal curvilinear coordinate systems.

When a scalar field, e.g., density = $\rho(\underset{\sim}{y},t)$, is differentiated, the curvature in the $\underset{\sim}{y}$ system adds no complications, and we simply have

$$\frac{\partial \rho}{\partial x_j'} = \frac{\partial y_k}{\partial x_j'} \frac{\partial \rho}{\partial y_k} = \frac{1}{h_{(j)}} \frac{\partial \rho}{\partial y_j} \, . \tag{2.4}$$

However, when a vector field, e.g., displacement = $u_i(\underset{\sim}{y},t)$, is differentiated, curvature in the $\underset{\sim}{y}$ system gives rise to additional terms which are developed in elementary texts on tensor calculus, see Spain (1960) or Washizu (1968). For the special case of orthogonal curvilinear systems we write

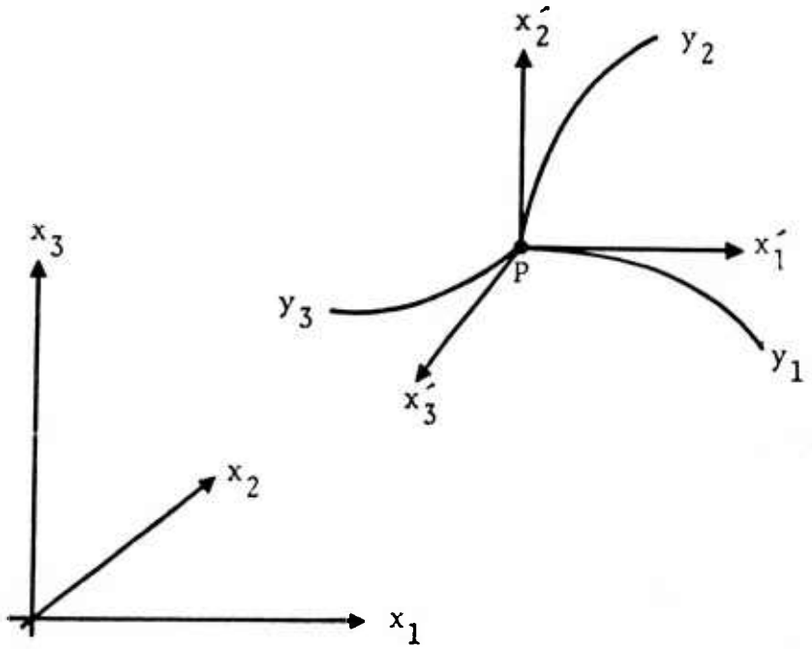


Fig. 2.3--Local Cartesian coordinate system centered at the point P.

TABLE 1

COMMON ORTHOGONAL COORDINATE SYSTEMS

Coordinate System	y_i	h_i	$\frac{\partial h_i}{\partial y_j}$
Cartesian	(x_1, x_2, x_3)	$(1,1,1)$	0
Cylindrical	(ρ, ϕ, z)	$(1,\rho,1)$	$\delta_{i2}\,\delta_{j1}$
Spherical	(r, θ, ϕ)	$(1,r,r\sin\theta)$	$\delta_{i2}\,\delta_{j1} + \delta_{i3}(\delta_{j1}\sin\theta + \delta_{j2}\, r\cos\theta)$
Parabolic	(ξ, η, z)	$\left(\sqrt{\xi^2+\eta^2}\,,\ \sqrt{\xi^2+\eta^2}\right),\ 1$	$\frac{(1-\delta_{i3})}{\sqrt{\xi^2+\eta^2}}(\xi\delta_{j1} + \eta\delta_{j2})$

$$\frac{\partial u_i}{\partial x'_j} = D_{ijk}\, u_k \tag{2.5}$$

with

$$D_{ijk} = \frac{\delta_{ik}}{h_{(j)}} \frac{\partial}{\partial y_j} - \frac{\delta_{jk}}{h_{(j)} h_{(i)}} \frac{\partial h_{(j)}}{\partial y_i} + \frac{\delta_{ij}}{h_{(i)} h_{(k)}} \frac{\partial h_{(i)}}{\partial y_k} . \tag{2.6}$$

Spatial derivatives of the discretized displacement field

$$u_i(\underset{\sim}{y},t) = \sum_{n=1}^{N} p_n(\underset{\sim}{y}) u_i(\underset{\sim}{Y}_n,t) \tag{2.7}$$

are conveniently expressed using the operator D_{ijk}:

$$\frac{\partial u_i}{\partial x'_j}(\underset{\sim}{y},t) = \sum_{n=1}^{N} D_{ijk}\, p_n(\underset{\sim}{y})\, u_k(\underset{\sim}{Y}_n,t) . \tag{2.8}$$

When $\underset{\sim}{y}$ denotes a Cartesian system (i.e., $\underset{\sim}{y} = \underset{\sim}{x}$ and $h_i = 1$), we note that D_{ijk} reduces to $\delta_{ik} \frac{\partial}{\partial x_j}$ and Eq. (2.8) reduces to our earlier expression, Eq. (2.2).

Strain, in the curvilinear, discretized space, is also conveniently expressed using the spatial operation

$$\varepsilon_{ij}(\underset{\sim}{y},t) = \frac{1}{2} \frac{\partial u_i}{\partial x'_j}(\underset{\sim}{y},t) + \frac{1}{2} \frac{\partial u_j}{\partial x'_j}(\underset{\sim}{y},t)$$

$$= \frac{1}{2} (D_{ijk} + D_{jik}) u_k(\underset{\sim}{y},t)$$

$$= \frac{1}{2} \sum_{n=1}^{N} (D_{ijk} + D_{jik}) p_n(\underset{\sim}{y}) u_k(\underset{\sim}{Y}_n,t) \tag{2.9}$$

Because the grid deforms with the medium in a Lagrangian formulation, we retain only the first order terms.

2.3.2 Inner-Element Interpolation

In the SWIS code, as in conventional finite element computer codes, spatial interpolation is defined element by element. Localized interpolation functions, which depend only on the geometry of a single element, characterize the spatial variations within the element so that

$$p_n(\underset{\sim}{y}) = p_n^e(\underset{\sim}{y})$$

for $\underset{\sim}{y}$ in V^e or on S^e where V^e and S^e denote the volume and the boundary surface of element e, respectively. The result is that $u_i(\underset{\sim}{y},t)$ in Eq. (2.7) depends only on the displacement of the node points bordering element e when $\underset{\sim}{y}$ is interior to or on the boundary of element e. A brief development of how interpolation functions are expressed for skewed element geometries is presented below.

Let us consider a local element transformation that serves to map skewed elements into a standard geometry, namely, a two-unit line segment in 1-D, a two-unit square in 2-D, and a two-unit cube in 3-D. We designate this natural element coordinate system $\underset{\sim}{z}$, illustrated for 3-D, in Fig. 2.2. The interpolation functions for a low order 3-D element are then expressed in the natural system as

$$p_m(\underset{\sim}{z}) = \frac{1}{8}(1 + z_1 Z_{1m})(1 + z_2 Z_{2m})(1 + z_3 Z_{3m})$$

in which m = 1, 2, ... 8 denotes the local node number for the element, illustrated in Fig. 2.2, and $Z_{im} = \pm 1$ denotes the $i^{\underline{th}}$ coordinate value for node m. The more general expression that applies in D-dimensional space is written

$$p_m(\underset{\sim}{z}) = \frac{1}{2^D} \prod_{i=1}^{D} (1 + z_i Z_{im}) \quad . \tag{2.10}$$

Spatial derivatives of the element interpolation functions are obtained directly from Eq. (2.10) to yield

$$\frac{\partial p_m(\underset{\sim}{z})}{\partial z_j} = \frac{Z_{jm}}{2^D} \prod_{\substack{i=1 \\ i \neq j}}^{D} (1 + z_i Z_{im}) \quad . \tag{2.11}$$

Interpolation functions for higher order elements are easily developed in the natural coordinate geometry, see for example, Frazier, et al., 1973.

We note that the interpolation functions expressed in natural element coordinates $\underset{\sim}{z}$ can be used directly for interpolating field variables within an element, e.g.,

$$u_i(\underset{\sim}{z},t) = \sum_{m=1}^{2^D} p_m(\underset{\sim}{z}) u_i(\underset{\sim}{Z}_m,t) \quad . \tag{2.12}$$

In addition, the same interpolation functions serve to express the mapping from natural element coordinates $\underset{\sim}{z}$ to the global curvilinear coordinates $\underset{\sim}{y}$, i.e.,

$$y_i(\underset{\sim}{z}) = \sum_{m=1}^{2^D} p_m(\underset{\sim}{z}) Y_{im} \tag{2.13}$$

and

$$\frac{\partial y_i(\underset{\sim}{z})}{\partial z_j} = \sum_{m=1}^{2^D} \frac{\partial p_m(\underset{\sim}{z})}{\partial z_j} Y_{im} \quad , \tag{2.14}$$

where Y_{im} denotes the $i^{\underline{th}}$ coordinate value of local node number m. Spatial derivatives of the interpolation functions with respect to global coordinates, Eqs. (2.2) and (2.8) then become

$$\frac{\partial p_m}{\partial y_i} = \left(\frac{\partial y_i}{\partial z_j}\right)^{-1} \frac{\partial p_m}{\partial z_j}$$

so that the derivative of the interpolated displacement field, Eq. (2.12), with respect to $\underset{\sim}{y}$ is computed using the expression

$$\frac{\partial u_i}{\partial y_j}(\underset{\sim}{z},t) = \left(\frac{\partial y_j}{\partial z_k}\right)^{-1} \sum_{m=1}^{2^D} \frac{\partial p_m}{\partial z_k}(\underset{\sim}{z})\, u_i(Z_m,t) \ . \qquad (2.15)$$

2.4 NOTATIONAL CONVENTION

The notation that has been adopted for the various spatial coordinate systems is illustrated in Figs. 2.2 and 2.3:

$\underset{\sim}{x}$ – global Cartesian coordinates.

$\underset{\sim}{y}$ – global orthogonal curvilinear coordinates.

$\underset{\sim}{x}'$ – local Cartesian coordinates aligned with the curvilinear system at the point $\underset{\sim}{x}' = 0$.

$\underset{\sim}{z}$ – natural element coordinates.

Capital symbols $\underset{\sim}{X}_n$, $\underset{\sim}{Y}_n$, and $\underset{\sim}{Z}_n$ are used to denote the spatial coordinate of a node point.

As illustrated by Eqs. (2.8) and (2.9), spatial derivatives in the various coordinate systems merely involve manipulations on the element interpolation functions. Subroutines handle these transformations in such a way as to minimize complications in the primary logic of the SWIS code. At the same time care has been taken to assure that excess calculations do not occur when employing a simple Cartesian system.

In the subsequent presentation, matrix notation will be used in preference to subscript notation for denoting arrays that arise from spatial discretization. This enables us to keep directional indices (subscript notation) separate from nodal indices (matrix notation). For convenience, all matrices are of order N, the total number of node points in the grid; and the symbols < >, { }, [ ], and [~] are used to denote a row, column, square, and diagonal matrix, respectively. Using this notation, Eq. (2.7) becomes

$$u_i(\underset{\sim}{y},t) = \langle p(\underset{\sim}{y}) \rangle \{U_i(t)\} \tag{2.16}$$

and the local Cartesian derivative of the spatially discretized displacement field is expressed

$$\frac{\partial u_i}{\partial x_j}(\underset{\sim}{y},t) = \langle D_{ijk}p(\underset{\sim}{y})\rangle \{U_k(t)\} \quad . \tag{2.17}$$

Note that we have used upper case to denote nodal displacement (a nodal subscript replacing the spatial argument), i.e.,

$$U_{in}(t) = u_i(\underset{\sim}{Y}_n,t) \tag{2.18}$$

or

$$\{U_i(t)\} = u_i(\underset{\sim}{Y}_n,t) \ , \quad n = 1, 2, \ldots N \ .$$

Finally, with regard to notation, we point out that the global arrays $\langle p(\underset{\sim}{y})\rangle$ and $\langle D_{ijk}p(\underset{\sim}{y})\rangle$ are never actually developed in the computing algorithm, but rather spatial interpolations are dealt with element by element. An array that is localized to a single element is denoted by a superscripted e, thus

$$\langle p(\underset{\sim}{y})\rangle = \sum_{e=1}^{E} \langle p^e(\underset{\sim}{y})\rangle$$

and

$$\langle D_{ijk}p(\underset{\sim}{y})\rangle = \sum_{e=1}^{E} \langle D_{ijk}p^e(\underset{\sim}{y})\rangle \ ,$$

E being the total number of elements in the global assemblage. Using this notation, $p_n^e(\underset{\sim}{y})$ is zero unless node n is associated with element e and $\underset{\sim}{y}$ lies within or on the boundary of element e.

2.5 CONSERVATION OF MOMENTUM

Conservation of momentum in a Lagrangian framework can be expressed by the virtual work expression

$$\int_V \left(\rho \ddot{u}_i \delta u_i + \sigma_{ij} \frac{\partial \delta u_i}{\partial x_j} - \bar{f}_i \delta u_i \right) dV - \int_{S_\sigma} \bar{\tau}_i \delta u_i dS = 0 \qquad (2.19)$$

in which u_i is the particle displacement, δu_i is a virtual displacement, $\ddot{u}_i$ is particle acceleration, σ_{ij} is stress, ρ is mass density, and $\bar{f}_i$ and $\bar{\tau}_i$ are specified body force and surface traction, respectively. S_σ is that portion of surface (internal or external) bounding the volume V to which tractions are applied. For general orthogonal curvilinear coordinates, the term $\partial \delta u_i / \partial x_j$ is taken to be $D_{ijk} \delta u_k$, the operator D_{ijk} having been defined above in Eq. (2.6). For a Cartesian system, the term simply becomes $\partial \delta u_i / \partial x_j$.

In the spatially discrete system conservation of momentum is expressed by substituting the interpolated displacement field from Eq. (2.16) into the virtual work expression above to obtain

$$\{\delta U_i\}^T ([M]\{\ddot{U}_i\} + \{R_i\} + \{Q_i\} - \{F_i\}) = 0 \qquad (2.20)$$

where

$$[M] = \sum_{e=1}^{E} \int_{V^e} \rho \langle p^e \rangle^T \langle p^e \rangle \, dV$$

$$\{F_i\} = \sum_{e=1}^{E} \int_{V^e} \bar{f}_i \langle p^e \rangle^T dV + \sum_{b=1}^{B} \int_{S_\sigma^b} \bar{\tau}_i \langle p^b \rangle^T \, dS$$

$$\{R_i\} = \sum_{e=1}^{E} \int_{V^e} \sigma_{jk} \langle D_{jki} p^e \rangle^T \, dV$$

$$\{Q_i\} = \sum_{e=1}^{E} \int_{V^e} q_{ij} \langle D_{jki} p^e \rangle^T \, dV$$

in which E is the total number of elements in the grid and B is the number of element surfaces with applied tractions. An artificial stress $q_{ij} = q_{ij}(\dot{\underset{\approx}{\varepsilon}})$ has been introduced for the purpose of damping spurious high frequency numerical oscillations.

The zero matrix is the only vector orthogonal to all possible (unconstrained) virtual displacements $\{\delta U_i\}$, therefore Eq. (2.19) yields a series of simultaneous equations expressing conservation of momentum node by node:

$$[M]\{\ddot{U}_i(t)\} + \{R_i(t)\} + \{Q_i(t)\} = \{F_i(t)\} \quad . \tag{2.21}$$

In contrast to conventional finite difference methods, free surfaces and loaded surfaces are "automatically" provided for in the above equations of motion through the forcing term $\{F_i(t)\}$. Node points with a specified displacement time history and node points along a transmitting boundary are not automatically handled by the virtual work expression, and therefore, these constrained node points require special considerations. For convenience, we have simply modified the definition of the forcing term at the constrained node points so that Eq. (2.21) applies, without exception, to all points in the grid. The modified prescription for $F_{in}(t)$ suited for the constraint condition at node n is given below in Eqs. (2.30) and (2.31).

In contrast to the conventional finite element method, we note that the constitutive properties have not been applied in the development of Eq. (2.21). The load-deformation properties of the material are introduced below at an intermediate stage in the time stepping scheme.

2.6 TIME STEPPING SCHEME

In selecting a time stepping scheme, we have considered the relative merits of explicit and implicit methods. In order to achieve satisfactory accuracy in the propagation of sharp wave fronts (wave length equal to 10 to 15 grid dimensions), a small time step is needed, roughly equal to that required for stability of an explicit method. When a numerical calculation involves such small time steps, explicit methods are strongly favored over implicit methods. Explicit methods generally require many times fewer operations per time step. The number of multiply and add operations for the implicit schemes used in finite element codes, such as SAP, increases as $N^{5/3}$ for cube-like blocks of 3-D meida; whereas for explicit methods the number of operations increases linearly with N, N being the total number of node points in the numerical grid. Furthermore, algorithms based on explicit methods are simpler to program and generally more flexible for introducing nonlinear material response behaviors. The simplicity of our explicit wave calculation scheme has made it possible to develop a very efficient parallel algorithm for processing linear seismic waves on the ILLIAC IV computer.

Stress, $\sigma_{ij}^{e}(\underset{\sim}{y}, t-\Delta t)$, $e = 1,2, \ldots E$; velocity, $\{\dot{U}_i(t - \frac{\Delta t}{2})\}$; displacement, $\{U_i(t)\}$; and node positions, $\{Y_i(t)\}$, are advanced in time by Δt using a four stage computing sequence.

Stage 1: Strain Rate

Compute strain rate for element e

$$\dot{\varepsilon}^e_{ij}\left(\underset{\sim}{y},t-\frac{\Delta t}{2}\right)=\frac{1}{2}\frac{\partial \dot{u}_i}{\partial x_j}\left(\underset{\sim}{y},t-\frac{\Delta t}{2}\right)+\frac{1}{2}\frac{\partial \dot{u}_j}{\partial x_i}\left(\underset{\sim}{y},t-\frac{\Delta t}{2}\right)$$

$$=\frac{1}{2}\left\langle D_{ijk}p^e(\underset{\sim}{y})+D_{jik}p^e(\underset{\sim}{y})\right\rangle\left|\dot{U}_k\left(t-\frac{\Delta t}{2}\right)\right| \tag{2.22}$$

where $\dot{\varepsilon}^e_{ij}\left(\underset{\sim}{y},t-\frac{\Delta t}{2}\right)$ is the strain rate evaluated at a discrete point within element e (strategic points in V^e for evaluating the $\{R_i\}$ integral of Eq. (2.20) which may, under special conditions, be confined to the centroid point $\bar{\underset{\sim}{y}}^e$). That is, the terms $\langle D_{ijk}p^e(\underset{\sim}{y})\rangle$ are evaluated at the integration points for element e.

Stage 2: Stress Rate and Stress

Compute stress rate for element e

$$\dot{\sigma}^e_{ij}\left(\underset{\sim}{y},t-\frac{\Delta t}{2}\right)=f\left(\dot{\underset{\approx}{\varepsilon}}^e,\ \underset{\approx}{\sigma}^e\right) \tag{2.23}$$

which, for linear isotropic material, becomes

$$\dot{\sigma}^e_{ij}\left(\underset{\sim}{y},t-\frac{\Delta t}{2}\right)=2\mu^e\,\dot{\varepsilon}^e_{ij}\left(\underset{\sim}{y},t-\frac{\Delta t}{2}\right)+\lambda^e\,\delta_{ij}\,\dot{\varepsilon}^e_{kk}\left(\underset{\sim}{y},t-\frac{\Delta t}{2}\right) \tag{2.24}$$

where μ^e and λ^e are Lamé's elastic constants for the material in element e. The stress at the advanced time is then computed by direct integration

$$\sigma^e_{ij}(\underset{\sim}{y},t)=\sigma^e_{ij}(\underset{\sim}{y},\ t-\Delta t)+\Delta t\,\dot{\sigma}^e_{ij}\left(\underset{\sim}{y},t-\frac{\Delta t}{2}\right). \tag{2.25}$$

Compute artificial stress q^e_{ij} which serves to damp spurious high frequency numerical oscillations

$$q^e_{ij}(\underset{\sim}{y},t) = \Delta t\ \beta^e \dot{\sigma}^e_{ij}\left(\underset{\sim}{y},t - \frac{\Delta t}{2}\right) \tag{2.26}$$

where $\beta^e \approx 0.15$ is a dimensionless damping coefficient. For the special case of linear waves (linear material and small displacements) the damping provided by the above expression for q^e_{ij}, with β uniform between elements, results in the damping of each natural mode of vibration (i.e., each eigenfunction of the linear system) as the square of the corresponding natural frequency (Frazier, et al., 1973). Also, we note that no damping occurs in regions of stationary stress. The isotropic component, q^e_{kk}, is equivalent to linear damping used in Lagrangian finite difference shock codes, see for example, Richtmyer and Morton, 1967.

Stage 3: Restoring Forces (equivalent to stress gradients)

Compute the nodal restoring forces that result from stresses in element e

$$\{R^e_i(t) + Q^e_i(t)\} = \int_{V^e} \langle D_{jki} p(\underset{\sim}{y})\rangle^T \left(\sigma^e_{jk}(\underset{\sim}{y},t) + q^e_{jk}(\underset{\sim}{y},t)\right) dV\ . \tag{2.27}$$

In many applications, a single integration point at the centroid of the element is sufficient for evaluating the integral. However, for cases in which the strain energy that is neglected by sampling only at the centroid becomes significant, stress is computed (Stages 1 and 2) at two points for each spatial dimension of the element, and in this way the spatial variations within an element are treated in the integration above. Further discussion on the treatment of inner element variations in stress is presented in Appendix A. As an incidental note, Frazier, et al., 1973, have shown that for rectilinear 3-D

grids, a one-point integration procedure is equivalent to the cell-centered-stress finite difference method that is commonly used in Lagrangian shock codes.

The integration above is performed for each element to obtain the restoring force of the medium on all the nodes in the grid

$$\{R_i(t) + Q_i(t)\} = \sum_{e=1}^{E} \{R_i^e(t) + Q_i^e(t)\} \ . \qquad (2.28)$$

Actually, the global arrays $\{R_i(t)\}$ and $\{Q_i(t)\}$ are not stored, but rather the effects of the element restoring forces are directly accumulated in the nodal acceleration calculations developed in Stage 4.

Stage 4: Motion of the Node Points

Nodal accelerations are computed directly from Eq. (2.21)

$$\{\ddot{U}_i(t)\} = [\tilde{M}]^{-1}\{F_i(t)\} - [\tilde{M}]^{-1}\{R_i(t) + Q_i(t)\} \qquad (2.29)$$

where the so-called lumped mass matrix $[\tilde{M}]$ is obtained by replacing each diagonal term in the distributed mass matrix $[M]$ by the sum of the terms in the row in which it appears. This operation yields a diagonal mass matrix thereby making the inversion of the mass matrix in Eq. (2.29) trivial.

Equation (2.29) directly applies to all unconstrained internal and boundary nodes, including internal nodes with applied body forces and boundary nodes with applied tractions (zero or otherwise). A modified prescription for $F_{in}(t)$ is used at constrained node points so that Eq. (2.29) is universally applicable to all node points in the grid.

The SWIS code accommodates two types of constraint conditions: The first type involves a constrained component of displacement in which displacement is made to follow a prescribed time history $\bar{u}_i(Y_{\sim n}, t)$. This condition is satisfied with

$$F_{in}(t) = R_{in}(t) + Q_{in}(t) + \frac{\tilde{M}_{nn}}{\Delta t^2}\,\bar{u}_i(Y_{\sim n}, t + \Delta t) - \frac{\tilde{M}_{nn}}{\Delta t^2}\,U_{in}(t) - \frac{\tilde{M}_{nn}}{\Delta t}\,\dot{U}_{in}\left(t - \frac{\Delta t}{2}\right) \tag{2.30}$$

where n denotes the node number.

The second type of node constraint involves a transparent boundary condition in which a boundary point is made to reflect almost no energy. In this case the nodal forcing term is set to

$$F_{in}(t) = \frac{1}{2}\,R_{in}(t) + \frac{1}{2}\,Q_{in}(t) - \frac{\tilde{M}_{nn}}{2\Delta t}\,\dot{U}_{in}\left(t - \frac{\Delta t}{2}\right) \tag{2.31}$$

Nodal velocities at the advanced time $t + \Delta t/2$ are computed by direct (numerical) integration of the nodal accelerations

$$\left\{\dot{U}_i\left(t + \frac{\Delta t}{2}\right)\right\} = \left\{\dot{U}_i\left(t - \frac{\Delta t}{2}\right)\right\} + \Delta t\left\{\ddot{U}_i(t)\right\} \tag{2.32}$$

and similarly, the nodal displacements are advanced in time

$$\{U_i(t + \Delta t)\} = \{U_i(t)\} + \Delta t\left\{\dot{U}_i\left(t + \frac{\Delta t}{2}\right)\right\} \tag{2.33}$$

and

$$\{Y_i(t + \Delta t)\} = \{Y_i(t)\} + \Delta t[\tilde{H}_{(i)}]^{-1}\left\{\dot{U}_i\left(t + \frac{\Delta t}{2}\right)\right\} \tag{2.34}$$

where $[\tilde{H}_{(i)}]$ is a diagonal listing of the $i^{\underline{th}}$ metric coefficient for each node in the grid. Equations (2.32), (2.33), and (2.34) are employed for all node points in the grid; no exception is made at this point for the calculations at constrained nodes.

The sequence of calculations outlined in the four stages above yields the necessary variables for continuing into the subsequent time step, i.e., set $t = t + \Delta t$ and return to Stage 1. Approximately 500 floating point multiply and add operations are required per 3-D element to advance the solution one time step using Cartesian coordinates and a one-point integration scheme in Stage 3. To put this number is perspective we note that roughly one-half of this effort would be required per node to multiply the non-zero terms in a 3-D finite element stiffness matrix by the nodal displacements (about 250 floating point multiply and add operations). Thus, the algorithm developed above should be exceedingly fast for both linear and nonlinear stress wave calculations. This conjecture is supported by ILLIAC test calculations, some of which are presented in Section IV.

2.7 CONSERVATION OF ENERGY

Kinetic energy, strain energy, dissipated energy (artificial viscosity), and load potential are computed at each time step based on the expressions:

$K(t)$ = kinetic energy at time t

$$= \frac{1}{2}\left\{\dot{U}_i\left(t + \frac{\Delta t}{2}\right)\right\}^T [M] \left\{\dot{U}_i\left(t - \frac{\Delta t}{2}\right)\right\} \tag{2.35}$$

$S(t)$ = strain energy (internal energy) at time t

$$= S(t-\Delta t) + \Delta t\, \dot{S}\left(t - \frac{\Delta t}{2}\right)$$

$$= S(t-\Delta t) + \frac{\Delta t}{2}\left\{\dot{U}_i\left(t - \frac{\Delta t}{2}\right)\right\}^T \left\{R_i(t) + R_i(t-\Delta t)\right\} \tag{2.36}$$

$D(t)$ = dissipated energy at time t resulting from artificial viscosity

$$= D(t-\Delta t) + \Delta t\, \dot{D}\left(t - \frac{\Delta t}{2}\right)$$

$$= D(t-\Delta t) + \frac{\Delta t}{2}\left\{\dot{U}_i\left(t - \frac{\Delta t}{2}\right)\right\}^T \left\{Q_i(t) + Q_i(t-\Delta t)\right\}$$

$$\cong D(t-\Delta t) + \Delta t\left\{\dot{U}_i\left(t - \frac{\Delta t}{2}\right)\right\}^T \left\{Q_i(t)\right\} \tag{2.37}$$

$L(t)$ = load potential (energy entering or leaving the system through the boundaries or through the action of body forces) at time t resulting from body forces and surface tractions (Eq. (2.20)), specified displacement time history (Eq. (2.30)), and transmitting boundaries (Eq. (2.31))

$$= L(t-\Delta t) + \Delta t\, \dot{L}\left(t - \frac{\Delta t}{2}\right)$$

$$= L(t-\Delta t) + \frac{\Delta t}{2}\left\{\dot{U}_i\left(t - \frac{\Delta t}{2}\right)\right\}^T \left\{F_i(t) + F_i(t-\Delta t)\right\} \qquad (2.38)$$

We note that an approximate expression is employed for estimating dissipated energy $D(t)$ in Eq. (2.37), because the array of dissipation forces $\{Q_i(t-\Delta t)\} - \beta\Delta t \left\{\dot{R}_i\left(t - \frac{3\Delta t}{2}\right)\right\}$, which is needed for a consistent calculation, is not retained in core at the advanced time t.

When SWIS is operated without artificial damping, energy is conserved in the calculations, i.e.,

$$L(t) - K(t) - S(t) = 0 + \text{computer round-off} \qquad (2.39)$$

with $D(t) = 0$. Conservation of energy has been observed for linear wave calculations. However, because the method has been formulated from an energy principle, Eq. (2.39) will also be satisfied for waves in nonlinear materials with $D(t) = 0$.

Energy conservation can be demonstrated directly from the discrete equations of motion. The two discrete equations — Eq. (2.21), centered at time t, and Eq. (2.21), centered at time $t - \Delta t$ — are averaged to yield an equation of motion at $t - \Delta t/2$

$$\frac{1}{2\Delta t}\,[M]\left\{\dot{U}_i\left(t + \frac{\Delta t}{2}\right) - \dot{U}_i\left(t - \frac{3\Delta t}{2}\right)\right\} + \frac{1}{2}\left\{R_i(t) + R_i(t-\Delta t)\right\}$$

$$+ \frac{1}{2}\left\{Q_i(t) + Q_i(t-\Delta t)\right\} = \frac{1}{2}\left\{F_i(t) + F_i(t-\Delta t)\right\}$$

in which Eq. (2.32) has been employed to replace $\{\ddot{U}_i(t)\}$ and $\{\ddot{U}_i(t-\Delta t)\}$ by

$$\frac{1}{\Delta t}\left\{\dot{U}_i\left(t + \frac{\Delta t}{2}\right) - \dot{U}_i\left(t - \frac{\Delta t}{2}\right)\right\}$$

and

$$\frac{1}{\Delta t}\left\{\dot{U}_i\left(t - \frac{\Delta t}{2}\right) - \dot{U}_i\left(t - \frac{3\Delta t}{2}\right)\right\}$$

respectively. This expression is then pre-multiplied by nodal velocities $\{\dot{U}_i(t - \Delta t/2)\}^T$ to obtain variations in energy over the time step Δt

$$\frac{1}{2}\left\{\dot{U}_i\left(t - \frac{\Delta t}{2}\right)\right\}^T [M]\left\{\dot{U}_i\left(t + \frac{\Delta t}{2}\right)\right\} - \frac{1}{2}\left\{\dot{U}_i\left(t - \frac{\Delta t}{2}\right)\right\}^T [M]\left\{\dot{U}_i\left(t - \frac{3\Delta t}{2}\right)\right\}$$

$$+ \frac{\Delta t}{2}\left\{\dot{U}_i\left(t - \frac{\Delta t}{2}\right)\right\}^T \left\{R_i(t) + R_i(t-\Delta t)\right\}$$

$$+ \frac{\Delta t}{2}\left\{\dot{U}_i\left(t - \frac{\Delta t}{2}\right)\right\}^T \left\{Q_i(t) + Q_i(t-\Delta t)\right\}$$

$$= \frac{\Delta t}{2}\left\{\dot{U}_i\left(t - \frac{\Delta t}{2}\right)\right\}^T \left\{F_i(t) + F_i(t-\Delta t)\right\} \quad .$$

Using the notation introduced in Eqs. (2.35) through (2.38) we have

$$K(t) - K(t-\Delta t) + S(t) - S(t-\Delta t) + D(t) - D(t-\Delta t) = L(t) - L(t-\Delta t) \quad . \tag{2.40}$$

Consequently, all of the energy in the system has been accounted for; changes in load potential are reflected by changes in kinetic energy, strain energy (internal energy), and dissipated energy. Beginning at some initial time t_0 with $K(0) + S(0) + D(0) = L(0)$, Eq. (2.40) is applied repetitively from t_0 to t to yield

$$K(t) + S(t) + D(t) = L(t) \tag{2.41}$$

with no dependence on the constitutive properties. We note that, for the case of a yielding material, not all of the strain energy is recoverable. Thus we see that S(t) contains both the recoverable strain energy and the internal plastic work.

2.8 SPECIAL CASES: CARTESIAN COORDINATES, RECTILINEAR GRIDS, AND LINEAR MATERIALS

The preceding formulation for time stepping nonlinear stress waves through 1-D, 2-D, and 3-D curvilinear geometries may involve procedures not commonly found either in finite element literature or finite difference literature. In an effort to make the presentation more easily understood, some special cases will be considered.

2.8.1 Cartesian Coordinates

The presentation of the computing scheme is complicated somewhat by the inclusion of general orthogonal curvilinear geometries. Therefore, we will summarize the time stepping procedure using Cartesian coordinates. This not only removes much of the abstractness from spatial derivatives in the discrete system but also enables us to focus on the key operations that are involved in completing a time step.

As described above in Section 2.6, stresses, velocities, displacements, and node positions are advanced in time using a four stage computing sequence. Denoting parameter initialization as stage 0, the following operations are performed:

0. Initialize Values

$$\left\{X_i(t)\right\} ; \left\{U_i(t)\right\} ; \left\{\dot{U}_i\left(t - \frac{\Delta t}{2}\right)\right\}; \sigma^e_{ij}(\underset{\sim}{x}, t - \Delta t) ,$$

$$e = 1,2, \ldots E; \; e = 1.$$

1. Compute Strain Rate

$$\dot{\varepsilon}^e_{ij}\left(\underset{\sim}{x},\ t - \frac{\Delta t}{2}\right) = \frac{1}{2}\left\langle\frac{\partial p^e(\underset{\sim}{x})}{\partial x_j}\right\rangle \left\{\dot{U}_i\left(t - \frac{\Delta t}{2}\right)\right\}$$

$$+ \frac{1}{2}\left\langle\frac{\partial p^e(\underset{\sim}{x})}{\partial x_i}\right\rangle \left\{\dot{U}_j\left(t - \frac{\Delta t}{2}\right)\right\} \qquad (2.42)$$

2. Compute Stress Rate and Stress

$$\dot{\sigma}^e_{ij}\left(\underset{\sim}{x},\ t - \frac{\Delta t}{2}\right) = f\left(\underset{\approx}{\dot{\varepsilon}}^e,\ \underset{\approx}{\sigma}^e\right) \cong \left[2\mu^e\dot{\varepsilon}^e_{ij}\left(\underset{\sim}{x},\ t - \frac{\Delta t}{2}\right)\right.$$

$$\left. + \lambda^e\delta_{ij}\dot{\varepsilon}^e_{kk}\left(\underset{\sim}{x},\ t - \frac{\Delta t}{2}\right)\right] \qquad (2.43)$$

$$\sigma^e_{ij}(\underset{\sim}{x},t) = \sigma^e_{ij}(\underset{\sim}{x},\ t - \Delta t) + \Delta t\ \dot{\sigma}^e_{ij}\left(\underset{\sim}{x},\ t - \frac{\Delta t}{2}\right) \qquad (2.44)$$

$$q^e_{ij}(\underset{\sim}{x},t) = \beta^e\Delta t\ \dot{\sigma}^e_{ij}\left(\underset{\sim}{x},\ t - \frac{\Delta t}{2}\right) \qquad (2.45)$$

3. Compute Restoring Forces

$$\left\{R^e_i(t) + Q^e_i(t)\right\} = \int_{V^e}\left\langle\frac{\partial p^e}{\partial x_j}(\underset{\sim}{x})\right\rangle^T\left(\sigma^e_{ij}(\underset{\sim}{x},t)\right.$$

$$\left. + q^e_{ij}(\underset{\sim}{x},t)\right)\ dV \qquad (2.46)$$

Steps 1 and 2 are repeated for each integration point $\underset{\sim}{x}$ located in element e. Steps 1, 2, and 3 are repeated for each element in the grid to yield

$$\left\{R_i(t) + Q_i(t)\right\} = \sum_{e=1}^{E}\left\{R^e_i(t) + Q^e_i(t)\right\}$$

4. Compute Motion

$$\left\{\ddot{U}_i(t)\right\} = [\tilde{M}]^{-1}\left\{F_i(t) - R_i(t) - Q_i(t)\right\} \tag{2.47}$$

with

$$F_{in}(t) = \sum_{e=1}^{E} \int_{V^e} \bar{f}_i \, p_n^e \, dV + \sum_{b=1}^{B} \int_{S_\sigma^b} \bar{\tau}_i \, p_n^b \, dS$$

for the case in which $\underset{\sim}{X}_n$ represent an unconstrained internal node or a boundary node with applied tractions (zero or otherwise),

$$F_{in}(t) = R_{in}(t) + Q_{in}(t) + \frac{\tilde{M}_{nn}}{\Delta t^2} \bar{u}_i(\underset{\sim}{X}_n, t + \Delta t) - \frac{\tilde{M}_{nn}}{\Delta t^2} U_{in}(t) - \frac{\tilde{M}_{nn}}{\Delta t} \dot{U}_{in}\left(t - \frac{\Delta t}{2}\right)$$

for the case in which $\underset{\sim}{X}_n$ is driven by the specified time history $\bar{u}_i(\underset{\sim}{X}_n, t)$, or

$$F_{in}(t) = \frac{1}{2} R_{in}(t) + \frac{1}{2} Q_{in}(t) - \frac{\tilde{M}_{nn}}{2\Delta t} \dot{U}_{in}\left(t - \frac{\Delta t}{2}\right)$$

for the case in which $\underset{\sim}{X}_n$ is positioned along a transmitting boundary.

$$\left\{\dot{U}_i\left(t + \frac{\Delta t}{2}\right)\right\} = \left\{\dot{U}_i\left(t - \frac{\Delta t}{2}\right)\right\} + \Delta t\left\{\ddot{U}_i(t)\right\} \tag{2.48}$$

$$\left\{U_i(t + \Delta t)\right\} = \left\{U_i(t)\right\} + \Delta t\left\{\dot{U}_i\left(t + \frac{\Delta t}{2}\right)\right\} \tag{2.49}$$

$$\left\{X_i(t + \Delta t)\right\} = \left\{X_i(t)\right\} + \Delta t\left\{\dot{U}_i\left(t + \frac{\Delta t}{2}\right)\right\} . \tag{2.50}$$

Time is advanced by Δt, e is set to one, and program control is returned to stage 1.

2.8.2 Rectilinear Grids

In general, the grid geometry will not be rectilinear. As discussed in Section 2.2.2 and illustrated in Fig. 2.2, elements that appear skewed in the global coordinate system $\underset{\sim}{y}$ are mapped into local element coordinates $\underset{\sim}{z}$ where each element appears as a cube (or a square in 2-D). Simple polynomial functions, Eq. (2.10), are then employed for interpolating spatial quantities over the interior of the element. Spatial derivatives with respect to the local element coordinate system are expressed in terms of derivatives of the interpolation functions, Eq. (2.11). Because the polynomial interpolation functions also express the transformation from local element coordinates to global problem coordinates, spatial derivatives of the interpolated field variables are evaluated at a specified point ($-1 \leq z_i \leq +1$) in an element through simple manipulations on the interpolation-function derivatives, Eq. (2.15).

The interpolation functions and their derivatives, expressed in local element coordinates, are the same for all elements in the grid. A subroutine has been constructed for producing the values of these spatial functions at any specified point $-1 \leq z_i \leq +1$. To proceed from these values and compute the value of a discretized field variable or its derivative at the specified point in the element merely involves a few matrix operations; consequently, the spatial differencing that is involved in the time stepping scheme, Eqs. (2.22) and (2.27), has not been expressed explicitly in the development. We do not concern ourselves with these details in actual code development. However, we will develop the particular spatial schemes that arise in 3-D rectilinear grids to indicate nature of the spatial discretization.

For the special case of a Cartesian global coordinate system, spatial interpolation in a skewed-brick 3-D element is expressed by

$$u_i(z,t) = \sum_{m=1}^{8} p_m^e(\underset{\sim}{z})\ U_{im}^e(t)$$

$$= \frac{1}{8}(1+z_1)(1+z_2)(1+z_3)U_{i1}^e$$

$$+ \frac{1}{8}(1-z_1)(1+z_2)(1+z_3)U_{i2}^e$$

$$+ \frac{1}{8}(1+z_1)(1-z_2)(1+z_3)U_{i3}^e$$

$$+ \ldots \frac{1}{8}(1-z_1)(1-z_2)(1-z_3)U_{i8}^e$$

where the second subscript on the nodal displacements denotes a local node number as designated in Fig. 2.2. The coordinate mapping from local element coordinates to global coordinates, i.e., $x_i = x_i(\underset{\sim}{z})$, is obtained by replacing $u_i(\underset{\sim}{z},t)$ and $U_{im}^e(t)$ in the above expression by $x_i(\underset{\sim}{z})$ and X_{im}^e, respectively.

When we restrict the 3-D element to a rectilinear brick geometry in which x_i is parallel to z_i, the coordinate mapping reduces to

$$x_i = z_i \frac{1}{2} \Delta x_{(i)}^e + \overline{X}_i^e \qquad (2.51)$$

where

$$\bar{X}_i^e = \frac{1}{8} \sum_{m=1}^{8} X_{im}$$

$$\Delta x_1^e = \left(X_{11}^e, X_{13}^e, X_{15}^e, \text{ or } X_{17}^e\right) - \left(X_{12}^e, X_{14}^e, X_{16}^e, \text{ or } X_{18}^e\right)$$

$$\Delta x_2^e = \left(X_{21}^e, X_{22}^e, X_{25}^e, \text{ or } X_{26}^e\right) - \left(X_{23}^e, X_{24}^e, X_{27}^e, \text{ or } X_{28}^e\right)$$

$$\Delta x_3^e = \left(X_{31}^e, X_{32}^e, X_{33}^e, \text{ or } X_{34}^e\right) - \left(X_{35}^c, X_{36}^e, X_{37}^e, \text{ or } X_{38}^e\right)$$

That is, $\bar{X}_i^e$ is the centroidal point and $\Delta x_{(i)}^e$ is the element dimension in the direction x_i. The transformation Jacobian, given by Eq. (2.14), then becomes

$$\frac{\partial x_i}{\partial z_j} = \frac{1}{2} \Delta x_{(i)}^e \, \delta_{ij} \tag{2.52}$$

and

$$\left(\frac{\partial x_i}{\partial z_j}\right)^{-1} = \frac{2}{\Delta x_{(i)}^e} \delta_{ij} \tag{2.53}$$

for the case of a simple brick element geometry.

We combine the spatial derivatives of the element interpolation functions with the transformation Jacobian above as indicated in Eq. (2.15) to obtain the spatial derivatives of the discretized displacement field

$$\frac{\partial u_i}{\partial x_j} = \frac{\partial x_j}{\partial z_k}^{-1} \sum_{m=1}^{8} \frac{\partial p_m}{\partial z_k} (\underset{\sim}{z}) \, U_i(Z_m, t) \; .$$

In particular we obtain

$$\frac{\partial u_i}{\partial x_1} = \frac{1}{4\Delta x_1^e}\Big[(1+z_2)(1+z_3)\left(U_{i1}^e - U_{i2}^e\right)$$

$$+ (1-z_2)(1+z_3)\left(U_{i3}^e - U_{i4}^e\right)$$

$$+ (1+z_2)(1-z_3)\left(U_{i5}^e - U_{i6}^e\right)$$

$$+ (1-z_2)(1-z_3)\left(U_{i7}^e - U_{i8}^e\right)\Big]$$

$$\frac{\partial u_i}{\partial x_2} = \frac{1}{4\Delta x_2^e}\Big[(1+z_1)(1+z_3)\left(U_{i1}^e - U_{i3}^e\right)$$

$$+ (1-z_1)(1+z_3)\left(U_{i2}^e - U_{i4}^e\right)$$

$$+ (1+z_1)(1-z_3)\left(U_{i5}^e - U_{i7}^e\right)$$

$$+ (1-z_1)(1-z_3)\left(U_{i6}^e - U_{i8}^e\right)\Big]$$

$$\frac{\partial u_i}{\partial x_3} = \frac{1}{4\Delta x_3^e}\Big[(1+z_1)(1+z_2)\left(U_{i1}^e - U_{i5}^e\right)$$

$$+ (1-z_1)(1+z_2)\left(U_{i2}^e - U_{i6}^e\right)$$

$$+ (1+z_1)(1-z_2)\left(U_{i3}^e - U_{i7}^e\right)$$

$$+ (1-z_1)(1-z_2)\left(U_{i4}^e - U_{i8}^e\right)\Big] \qquad (2.54)$$

At the element center, z = 0, we obtain the familiar difference equation

$$\frac{\partial u_i}{\partial x_i} = \frac{1}{4\Delta x_1^e}\left(U_{i1}^e + U_{i3}^e + U_{i5}^e + U_{i7}^e - U_{i2}^e - U_{i4}^e - U_{i6}^e - U_{i8}^e\right) \tag{2.55}$$

Similar expressions are obtained for $(\partial u_i/\partial x_2)$ and $(\partial u_i/\partial x_3)$ at the centroid of the brick-shaped element.

2.8.3 Linear Materials

An important class of problems that involve small amplitude seismic waves can be treated using a linearly elastic material model. We express linear material behavior in the stress-strain relationship

$$\sigma_{ij} = c_{ijk\ell}\epsilon_{k\ell}$$

for the general case of anisotropic and heterogeneous media. With this restriction, Stages 1, 2 and 3 (Eqs. (2.22) through (2.28)) combine to yield

$$\{R_i(t)\} = [K_{ij}]\{U_j(t)\} \tag{2.56}$$

and

$$\{Q_i(t)\} = \beta\Delta t[K_{ij}]\{\dot{U}_i(t)\} \tag{2.57}$$

where

$$[K_{ij}] = \frac{1}{2}\sum_{e=1}^{E}\int_{V^e}\left\langle D_{k\ell i}p^e\right\rangle^T c_{k\ell mn}\left\langle D_{mnj}p^e\right\rangle dV \tag{2.58}$$

in which the indices i, j, k, ℓ, m, and n vary from one through the number of spatial dimensions. By restricting the development to Cartesian coordinates, the expression for the stiffness matrix can be reduced somewhat

$$[K_{ij}] = \frac{1}{2} \sum_{e=1}^{E} \int_{V^e} \left\langle \frac{\partial p^e}{\partial x_k} \right\rangle^T c_{ikj\ell} \left\langle \frac{\partial p^e}{\partial x_\ell} \right\rangle dV \tag{2.59}$$

The equation of motion for the discrete system, Eq. (2.21), then becomes

$$[M]\{\ddot{U}_i(t)\} + \beta\Delta t[K_{ij}]\left\{\dot{U}_i\left(t - \frac{\Delta t}{2}\right)\right\} + [K_{ij}]\{U_i(t)\} = \{F_i(t)\} \tag{2.60}$$

and the resulting time stepping algorithm (Stages 1, 2, 3 and 4) is expressed in a single equation,

$$\begin{aligned}\{U_i(t+\Delta t)\} = {} & \Delta t^2[\tilde{M}]^{-1}\{F_i(t)\} + 2\{U_i(t)\} \\ & - \{U_i(t-\Delta t)\} - \Delta t^2[\tilde{M}]^{-1}[K_{ij}]\{(1+\beta)U_j(t) \\ & - \beta U_j(t-\Delta t)\}\end{aligned} \tag{2.61}$$

The linear SWIS code, which is described in Section 3.2, is based on this linearized time stepping procedure.

III. ILLIAC CODES: DESIGN AND DEVELOPMENT

3.1 OVERVIEW

In order to implement a numerical code on the ILLIAC IV, two unique aspects of the machine require special consideration. The first is its parallel architecture. In designing a stress wave code, our approach has been to formulate numerical operations particularly suited to the parallel nature of the machine rather than to adapt an existing code with numerics designed for a conventional serial computer. The resulting algorithm proved relatively easy to write and debug on the ILLIAC.

The second aspect to be dealt with is the method of input and output. As the normal access is via the ARPANET over large geographical distances, we must use new procedures in program development and debugging. Program source and input data must be delivered to the machine by teletype or by file transfer from another host computer on the Net. Output must return again via teletype or by file transfer to a host with line printers. Even with the availability of printer output, interpreting the results from large 3-D numerical simulations, for example, can be exceedingly tedious without facilities for graphical display.

Our first attempt at running programs in this environment occurred in March 1973. At that time, the only services available at the ILLIAC IV host were text editing and a minimal mechanism for ILLIAC program submission. It took nearly one month to get results from our first run. The results were contained in a memory dump listing which was received by mail.

Since then, the capabilities of the ILLIAC system have evolved rapidly. The ARPANET File Transfer Protocol allows us to transfer programs from UCSD to Sunnyvale and obtain

line printer listings of ILLIAC dump files. With UCSD only a few miles from S^3, we can obtain substantial output from an ILLIAC run only an hour after program completion on the ILLIAC. The display software enables an ILLIAC program to output selected quantities under program control. This eliminates the tedious task of searching memory dumps for computed results. Since September 1973, the ILLIAC installation has maintained a job turn-around of roughly once or twice a day for two weeks out of every month. With new services and improved job turn-around, the ease of getting work done has improved enormously.

S^3's first ILLIAC code, a linearized version of SWIS, became operational in July 1973, and has been exercised on several modest problems. A more general nonlinear version of SWIS, presently about 1200 lines of GLYPNIR coding in size, is now in the testing stage. As a result of improved job turn-around, this new SWIS code has progressed from design stage to successful test runs in slightly less than three months. During that period, ILLIAC reliability was sufficient to debug the new code directly on the machine rather than simulate the ILLIAC with SSK as was often necessary before September. We estimate that our recent code design and debug rate on the ILLIAC has been nearly 50 percent of the rate at which we could have developed comparable code on S^3's 1108 computer. We estimate that roughly half of the ILLIAC development is spent communicating with the ILLIAC host site via an interactive terminal.

In conclusion, we have had satisfactory results from our first attempts to use the ILLIAC IV. Part of this success stems from careful selection of the algorithms we first tried to implement. Code development progressed relatively smoothly, though was inhibited somewhat by the effort of interactive communication with the ILLIAC facility. Further

attention must be given to the difficulties of handling large quantities of output before major calculation can be performed.

3.2 LINEAR STRESS WAVE CODE

3.2.1 General Description

The goal in our ILLIAC code development effort has been to numerically simulate stress waves in 3-D geologic materials. The first step in attaining this goal was to develop a time stepping algorithm for propagating small amplitude waves in linear materials. The linear algorithm is formulated in Section 2.8.3; the algorithm is expressed by Eq. (2.61). In limiting our attention to linear material, we were able to construct a compact, yet versatile, code that eased our first efforts to use the ILLIAC IV.

The algorithm accommodates nonsystematic node numbering of 1-D, 2-D, or 3-D numerical grids. As there is no relationship between grid numbering and the number of PE's (processing elements) in the array, very irregular grids consisting of beams, plates, and so forth, may be analyzed. Furthermore, the algorithm is as efficient for irregular grids as it is for systematic grids. This is accomplished by a work-ahead procedure in which PE's simultaneously compute contributions to the advanced displacements of several different nodes. A serial machine requires 261 floating point multiply and add operations to obtain a nodal displacement at the advanced time step in a 3-D grid (Eq. (2.61)). The ILLIAC algorithm performs an average of 4 parallel floating point operations plus 3 row sums for each advanced nodal displacement. Thus, only a small overhead has been added to accommodate the parallel operations.

Based on predicted computing rates for the ILLIAC, it appears that a carefully coded version of the linear time stepping algorithm should run I/O bound on the array. This would require a processing rate of roughly 0.4 seconds per time step for a 10,000 node 3-D grid. Accurate timings are not available for the present operational version of the linear code (programmed in GLYPNIR). However, it appears that our present computing rates are considerably slower than the theoretical rate noted above.

3.2.2 Numerical Problem Definition

The definition of a complex grid can require a large amount of data. In the most general case, the definition must include data for individually locating each node and some information about the interconnectivity of the elements in the grid. In addition, inhomogeneous material properties must be specified element by element throughout the grid. In all, about 17N data items are required to completely describe a totally arbitrary N-node 3-D grid. Such a requirement would make 3-D grids excessively tedious to set up.

In order to make the linear version of SWIS as flexible as possible and not tie it down to any particular grid generation scheme or finite element scheme, the influence coefficients for the grid are generated separately and become data for the time stepping algorithm. One limitation of this approach is that the algorithm can compute stress waves only in materials with linear stress-strain laws, since no provision is made for recomputing the influence coefficients during the time stepping. However, since the grid need only be generated once, one may employ as sophisticated a grid generation scheme as desired involving curved grids or structural appendages, with no effect on the

efficiency and speed of the time stepping algorithm.

In the present code configuration, spatial discretization is carried out on a serial computer using a conventional finite element code. This step produces the terms $\{F_i(t)\}$, $[\tilde{M}]$, and $[K_{ij}]$ of Eq. (2.61). These terms are then sorted by the serial machine program into the order required by the ILLIAC time stepping scheme. A sort algorithm has been designed to perform this sort on the ILLIAC (Frazier, et al., 1973), but has not been implemented as our recent work has turned to implementation of the nonlinear version of SWIS (described in Section 3.3).

At each time step, Eq. (2.61) is processed. It has the form

$$\{\underset{\sim}{U}(t+\Delta t)\} = \{\underset{\sim}{V}(t)\} + [\underset{\approx}{A}]\ \{\underset{\sim}{U}^d(t)\} \tag{3.1}$$

where

$$\{\underset{\sim}{U}^d(t)\} = (1+\beta)\{\underset{\sim}{U}(t)\} - \beta\{\underset{\sim}{U}(t-\Delta t)\} \tag{3.2}$$

$$\{\underset{\sim}{V}(t)\} = \Delta t^2[\tilde{M}]^{-1}\{\underset{\sim}{F}(t)\} + 2\{\underset{\sim}{U}(t)\} - \{\underset{\sim}{U}(t-\Delta t)\} \tag{3.3}$$

$$[\underset{\approx}{A}] = -\Delta t^2[\tilde{M}]^{-1}[\underset{\approx}{K}] \tag{3.4}$$

Underscores are used here in place of directional component subscripts i and j. Using nodal subscripts, Eq. (3.1) becomes

$$\underset{\sim}{U}_n(t+\Delta t) = \underset{\sim}{V}_n(t) + \sum_m \underset{\approx}{A}_{nm}\ \underset{\sim}{U}^d_m(t) \quad . \tag{3.5}$$

The terms $\underset{\sim}{U}_n$, $\underset{\sim}{V}_n$, $\underset{\sim}{U}^d_n$, and $\underset{\sim}{F}_n$ correspond to node n in the numerical grid and each represents 3 floating point numbers on the computer for a 3-D grid. Space is provided for each vector by storing sequentially across PE's. For example,

the 64th term of $\{\underset{\sim}{U}(t)\}$, corresponding to the displacement of the 64th node point in the grid, falls into PE 63 (Fig. 3.1). The vectors wrap around in core so that $\underset{\sim}{U}_{65}$ appears in PE 0. In general, vector term n appears in PEk where k is the remainder of dividing n-1 by 64. In the present version of linear SWIS, these vectors are core contained. A three-dimensional grid of N nodes would require 3N storage locations for each vector. With roughly 131K words of PE memory available, the code is limited to 3-D problems containing no more than N = 10,000 nodes.

The matrix of influence coefficients, $[\underset{\approx}{A}]$, is an N × N matrix of 3 × 3 submatrices, or 3N × 3N. For a problem of $N = 10^4$ nodes, the matrix would consume roughly 10^9 words of storage. However, $[\underset{\approx}{A}]$ is sparse with each row generally containing no more than 27 non-zero 3 × 3 submatrices. This is a consequence of the connectivity in a 3-D grid of skewed brick elements in which each interior node point is connected directly to 26 neighbor nodes and each boundary node to less than 26 neighbors. If the matrix is compressed to remove zero submatrices, the storage is reduced to roughly: (number of nodes in the grid) × (number of neighbors plus one) × (words of storage for a submatrix) = $10^4 \times 27 \times 10 = 2.7 \times 10^6$ words for $N = 10^4$. The extra word of storage for the 3 × 3 submatrix contains the row and column position of the submatrix in the uncompressed matrix.

These submatrices are stored on the high speed disk in an order appropriate for the sparse matrix multiply which is performed repetitively during the time stepping. As illustrated in Fig. 3.2, the column number of the nonzero terms in $[\underset{\approx}{A}]$ provides the PE destination, i.e., $\underset{\approx}{A}_{nm}$ is to appear in PEk where k is the remainder of the division of m-1 by 64. This scheme assures that, as the influence coefficients are read into PE memory from the I4 disk, each coefficient

DESTINATION OF NODAL DISPLACEMENTS IN PE MEMORY

PE0	PE1	PE2		PE62	PE63	PE0		PEk
$\underset{\sim}{U}_1$,	$\underset{\sim}{U}_2$,	$\underset{\sim}{U}_3$	...	$\underset{\sim}{U}_{63}$,	$\underset{\sim}{U}_{64}$,	$\underset{\sim}{U}_{65}$	...	$\underset{\sim}{U}_N$

DISPLACEMENTS IN PE MEMORY

PE0	PE1	PE2	...	PE63
$\underset{\sim}{U}_1$	$\underset{\sim}{U}_2$	$\underset{\sim}{U}_3$	...	$\underset{\sim}{U}_{64}$
$\underset{\sim}{U}_{65}$				$\underset{\sim}{U}_{128}$
.				.
.				.
.				.

Fig. 3.1--Schematic illustrating the arrangement of nodal displacements in PE memory.

DESTINATION OF NONZERO INFLUENCE COEFFICIENTS IN PE MEMORY

$$[\underset{\approx}{A}] = \begin{bmatrix} \text{PE0} & \text{PE1} & \text{PE2} & \text{PE3} & \cdots & \text{PE0} & \text{PE1} & \text{PE2} & \cdots \\ \underset{\approx}{A}_{11} & \underset{\approx}{A}_{12} & 0 & 0 & \cdots & 0 & \underset{\approx}{A}_{1,66} & \underset{\approx}{A}_{1,67} & \cdots \\ \underset{\approx}{A}_{21} & \underset{\approx}{A}_{22} & \underset{\approx}{A}_{23} & & & 0 & 0 & \underset{\approx}{A}_{2,67} & \cdots \\ 0 & \underset{\approx}{A}_{32} & & & & & & & \\ 0 & 0 & & & & & & & \\ \vdots & & & & & & & \vdots & \\ & & & & & & & \underset{\approx}{A}_{N,N} & \end{bmatrix}$$

INFLUENCE COEFFICIENTS IN PE MEMORY

PE0	PE1	PE2	...	PEk	...
$\underset{\approx}{A}_{11}$	$\underset{\approx}{A}_{12}$	$\underset{\approx}{A}_{1,67}$		$\underset{\approx}{A}_{1,k}$	
$\underset{\approx}{A}_{21}$	$\underset{\approx}{A}_{1,66}$	$\underset{\approx}{A}_{2,3}$			
		$\underset{\approx}{A}_{2,67}$		$\underset{\approx}{A}_{n,k}$	
$\underset{\approx}{A}_{m,65}$					
				$\underset{\approx}{A}_{n,k+64}$	

Fig. 3.2--Schematic illustrating the arrangement of the influence coefficients $\underset{\approx}{A}_{nm}$ in PE memory.

will arrive in the PE that contains the corresponding nodal displacement. For a 3-D problem, 9 parallel multiplications can then be performed for each 10 rows of influence coefficients that arrive from the disk. The only PE interactions that are needed, even for irregular grid configurations, result from summing accumulated products between PE's – one row sum per row of the sparse matrix $[\underset{\approx}{A}]$. More detail on our sparse matrix multiply scheme is presented in the following section and in Appendix II.

3.2.3 Time Stepping

The time stepping process consists of the calculation of Eq. (3.5) for each time step. The first term $\{\underset{\sim}{V}(t)\}$ of Eq. (3.5) involves column vector operations which require no interaction amongst ILLIAC PE's. As a result, it is easily computed in a parallel process. Similar column vector operations are involved in the calculation of $\{\underset{\sim}{U}^d(t)\}$. The significant calculation is the multiplication of the vector $\{\underset{\sim}{U}^d(t)\}$ by the large sparse matrix $[\underset{\approx}{A}]$. This multiplication accounts for almost all of the computation time that is required to complete one numerical time step. A sophisticated but simple mechanism has been developed to perform the sparse matrix multiply in parallel (Frazier, et al., 1973). The non-zero terms of $[\underset{\approx}{A}]$ in Eq. (3.5) are arranged on disk so that each 3×3 submatrix $\underset{\approx}{A}_{nm}$ arrives in the PE containing $\underset{\sim}{U}^d_m$. Furthermore, as successive terms of $[\underset{\approx}{A}]$ are read from disk the matrix row numbers n increase monotonically (but not necessarily sequentially) in each PE. This is done so that the sparse matrix multiply can be completed in the order of ascending row number.

The first submatrix $\underset{\approx}{A}_{nm}$ to arrive in each PE from the disk (the $\underset{\approx}{A}_{nm}$ with the lowest row number n that appears in each PE) is multiplied by the three-component vector

$\underset{\sim}{U}^d_m$ and the results are accumulated in a buffer $\underset{\sim}{R}$ along with the row number identifier n. This operation allows some PE's to work on the same row number n while other PE's work ahead on other row numbers. Since several rows may be processed simultaneously, a look-ahead buffer $\{\underset{\sim}{R}\}$ is maintained in each PE which contains both the elements $\underset{\sim}{R}$ and the row number n. Since rows will continuously be completed as new ones are started, $\{\underset{\sim}{R}\}$ need only be large enough to contain the maximum number of $\underset{\sim}{R}$'s to be worked on at one time in any given PE. On the average, all of the multiplies for $64/27 \simeq 2.4$ rows of the sparse matrix multiply are completed after such an operation. Rows that correspond to boundary nodes require less calculation.

During the matrix multiply, a test is made to see if all contributions from the sparse matrix multiply are ready to be summed for the node n_0. If all of the row numbers n from the submatrix multiply are greater than n_0, then all contributions for n_0 are completed (all PE's are now working on contributions to higher node numbers). The contributions for n_0 are then summed and added to the other terms in Eq. (3.5) to obtain the advanced nodal displacements $\underset{\sim}{U}_{n_0}(t+\Delta t)$. This displacement vector is stored in PEk, k being the remainder of n_0-1 divided by 64. If the contributions from row n_0+1 are completed, then node n_0+1 is also advanced in time by summing contributions from participating PE's, otherwise the next submatrix multiply in line for each PE is performed. The parallel submatrix multiplies, row sums, and disk reads continue until all of the $[\underset{\approx}{A}]$ matrix has been processed and all nodes have been advanced in time. The entire operation is repeated for each time step. (A more detailed description of the sparse matrix multiply appears in Appendix B.)

3.2.4 Tests

The first version of linear SWIS was coded in GLYPNIR and also in ASK. Several tests were made with the GLYPNIR code on the ILLIAC IV simulator at UCSD (Frazier, et al., 1973). The GLYPNIR version of linear SWIS with a simple grid generator became operational on the ILLIAC IV in April 1973. Because of difficulties with the disk hardware at that time, SWIS was run with the $[\underset{\approx}{A}]$ matrix held entirely in core. Several test runs involving the propagation of planar P waves in 3-D media have been made to check boundary conditions in the code. Successful runs of over 100 seconds on the ILLIAC were completed.

3.3 LAGRANGIAN STRESS WAVE CODE

3.3.1 Grid Configuration

A flexible scheme, described in Section II, has been devised for numerically simulating stress waves in geologic materials. The flexibility for handling highly irregular grid configurations has been compromised somewhat in adapting the scheme to the ILLIAC.

The general numerical scheme admits mixed element types (e.g., trahedra and hexahedra) with nonsystematic node and element numbering. Because of the difficulties in transferring information between PE's in an arbitrary manner on the ILLIAC, we associate grid cross-sections with PE's, as illustrated in Fig. 3.3. Adjacent grid cross-sections are associated with adjacent PE's so that points which are adjacent in the grid appear in the same or adjacent PE's.

The first grid dimension, which is normal to grid section mentioned above, is strung across PE's. This enables a string of 64 elements, lying in 64 contiguous cross-sections, to be processed in parallel. We note that the first grid dimension does not necessarily correspond to the first problem coordinate. To insure totally parallel operations for the bulk of the calculations on the ILLIAC, the requirement is made that each element string in the first grid dimension contains elements of the same basic type. That is, one element string cannot contain both tetrahedral and hexahedral (skewed-brick) elements. The present operating version of SWIS treats only one element type over the entire grid: 8-node hexahedra in 3-D, 4-node quadralaterals in 2-D, and 2-node line segments in 1-D. This restriction may be lifted in the future to allow for varying element types between element strings.

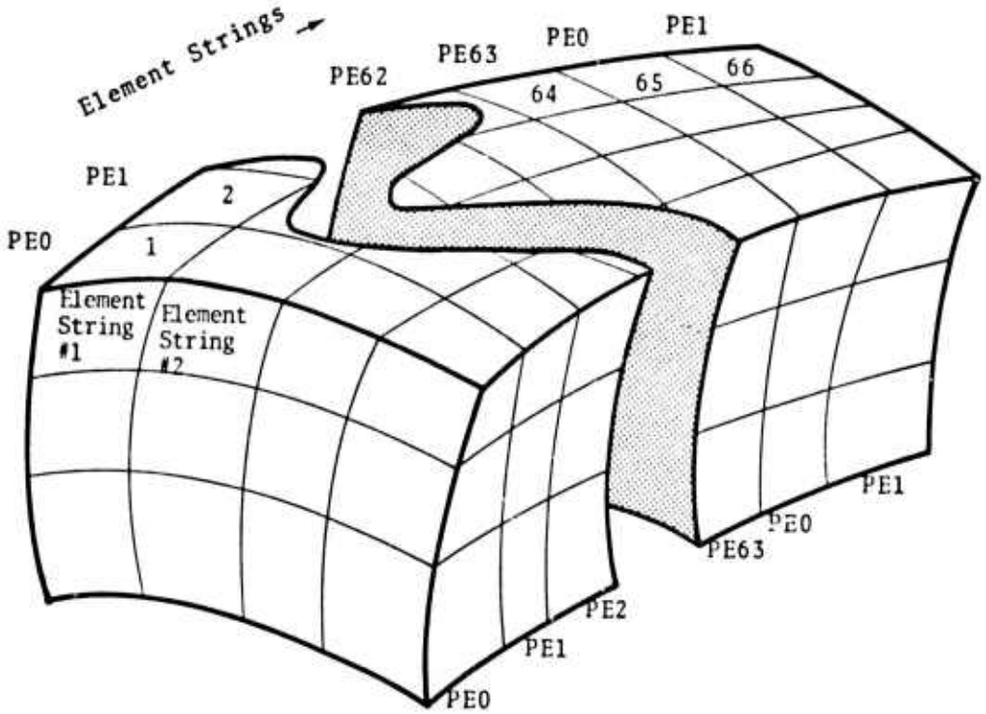


Fig. 3.3--Relationship between grid geometry and PE storage.

The requirement for uniformity of element types imposes restrictions only on the connectivity between elements, not on the size or shape of the elements. No restrictions are placed on material types, applied loads, or boundary conditions in any dimension of the grid. The ILLIAC SWIS code can also treat irregular geometric shapes. The number of elements can vary between node strings due to irregular grid boundaries, either internal or external. However, inefficiencies in PE utilization occur when the number of elements in a node string is not a multiple of 64.

3.3.2 Storage Scheme

The ILLIAC SWIS code uses three types of data storage:

1. General problem descriptors require only minor storage and include such terms as curvilinear coordinate designator, number of spatial dimensions, material descriptions, grid descriptions, time stepping data, and data for printing selected results.

2. Global storage contains nodal and element information for the entire grid. For each node point in the grid, the present version of SWIS stores coordinates (position), displacement, velocity, acceleration, boundary condition type, applied force (or displacement), and concentrated mass. Material type and element stress are stored for each element in the grid. Nodal and element storage are combined, storing one node with one element, to yield 9, 16, and 24 words of storage per 1-D, 2-D, and 3-D element, respectively. Figure 3.4 illustrates the various grid terms that appear in global storage.

The global storage scheme has been designed first to minimize interactions between PE's and second to simplify

STORAGE OF GRID QUANTITIES ACROSS PE'S

	PE0	PE1	... PEk ...
Element String 1	1	2	... k+1 ...
	65	66	... k+65 ...
Element String 2	1	2	... k+1 ...
	65	66	
	.	.	
	.	.	
	.	.	

	PEk k+1	Words of Storage
Element k+1 of Element String 2	Node Displacement	D
	Node Velocity	D
	Node Acceleration	D
	Node Coordinates	D
	Boundary Condition Type	1
	Applied Force (or Displacement)	D
	Node Mass	1
	Material Type	1
	Element Stress	D(D+1)/2

Words of storage per element = $(D^2 + 11D)/2 + 3$

where D is the number of spatial dimensions.

Fig. 3.4--Multiplexed storage of global variables.

disk accessing. As indicated in the previous section, global storage for one dimension of the grid (the first dimension) is strung across PE's with variables for cross-section n appearing in PEk where k is the remainder from the division of n-1 by 64. Figure 3.3 illustrates the relationship between the PE's and the grid configuration. All of the global variables needed to process one element, i.e., to compute element restoring forces, are always contained in two adjacent PE's (PE0 is considered to be adjacent to PE63). Node variables are routed left at the beginning of the loop for processing a string of 64 elements, and the computed element restoring forces are routed right at the end of the loop. Thus, by associating global storage with grid configuration, we have reduced PE interactions to simple, predictable routing operations.

Primarily for the purpose of expediting efficient use of the I4 disk, the various node and element quantities are multiplexed in global storage, as illustrated in Fig. 3.4. In the present design, global arrays will appear sequentially on the I4 disk with grid dimension one most rapidly varying in the grid numbering scheme. Thus, if a particular global variable is located at position n on the disk then the corresponding variable for the next higher element number would be located at position $n + (D^2 + 11D)/2 + 3$ where D is the number of spatial dimensions in the grid and $(D^2 + 11D)/2 + 3$ is the number of terms per element multiplexed into global storage.

3. Element string storage, which occupies about 60 words of fixed storage per PE, contains variables for processing one string of 64 elements. The element string storage serves as buffer storage for certain global variables (node positions and node velocities) and as storage for intermediate variables not stored globally (curvilinear coordinate metrics,

element interpolation functions and their spatial derivatives, element strain rates, element stress rates, and element restoring forces). By setting aside special element storage, we have managed to isolate inner PE exchanges from the major calculation phase of the code.

3.3.3 Initialization

As described in Section 2.2, a stress wave calculation is initiated by data that specifies the character of the grid, the (time-varying) boundary conditions, the material properties, the time stepping data, the initial conditions, and the type of results to be printed. In the present configuration, about 50 words of data are required to initiate a 3-D calculation. Approximately half of these data serve to define the grid. Nonzero initial conditions and time varying forcing terms require additional data.

Before proceding with the time stepping calculations, global storage is initialized. Based on the number of dimensions in the grid (and on the number stress components to be included in non-Cartesian calculations), the global multiplexed storage is dynamically assigned within a large segment of available core at run time. Displacements, velocities, and element stresses are set to a default value of zero. Boundary conditions and material types are set from the problem input data. A code-contained grid generator serves to produce node positions in both Cartesian and curvilinear grids. Nodal masses and nodal forces are computed by numerical integration, Eq. (2.20). The nodal accelerations are initialized to

$$\{\ddot{U}\} = [\tilde{M}]^{-1}\{F_i\} \quad ,$$

Eq. (2.29) with $\{R_i\} = \{Q_i\} = 0$. We note that, with the

exception of boundary conditions and nodal forces, initialization calculations are conveniently processed in parallel.

3.3.4 Time Stepping

The calculations that are performed to numerically propagate stress waves are described in Section 2.6, Eqs. (2.22) – (2.34). Figure 3.5 illustrates how these calculations are performed on the ILLIAC using the following basic operations: Grid positions and nodal velocities combine to yield element strain rates. The element strain rates combine with material properties to yield element stress rates, which are used to update total element stresses. Element restoring forces, which are computed from the element stresses, are combined with externally applied forces (or other boundary conditions) to produce nodal accelerations. Nodal displacements and nodal velocities are then advanced one time step by direct numerical integration of the nodal accelerations. This time stepping procedure continues until the wave simulation is completed.

We note that the algorithm has been organized so that interactions between PE's occur at only two points in the computing sequence. The initial operation in processing a string of 64 elements involves a parallel route left. This serves to bring global values of nodal velocity and nodal coordinates into local element storage. The nodal restoring forces for the 64-element string are then computed in parallel with no inner PE communications. These operations represent the bulk of calculations for completing one time step.

The final operation in processing a string of 64 elements involves a route right. The nodal restoring forces are divided by the corresponding nodal masses and accumulated in global storage as contributions to nodal accelerations.

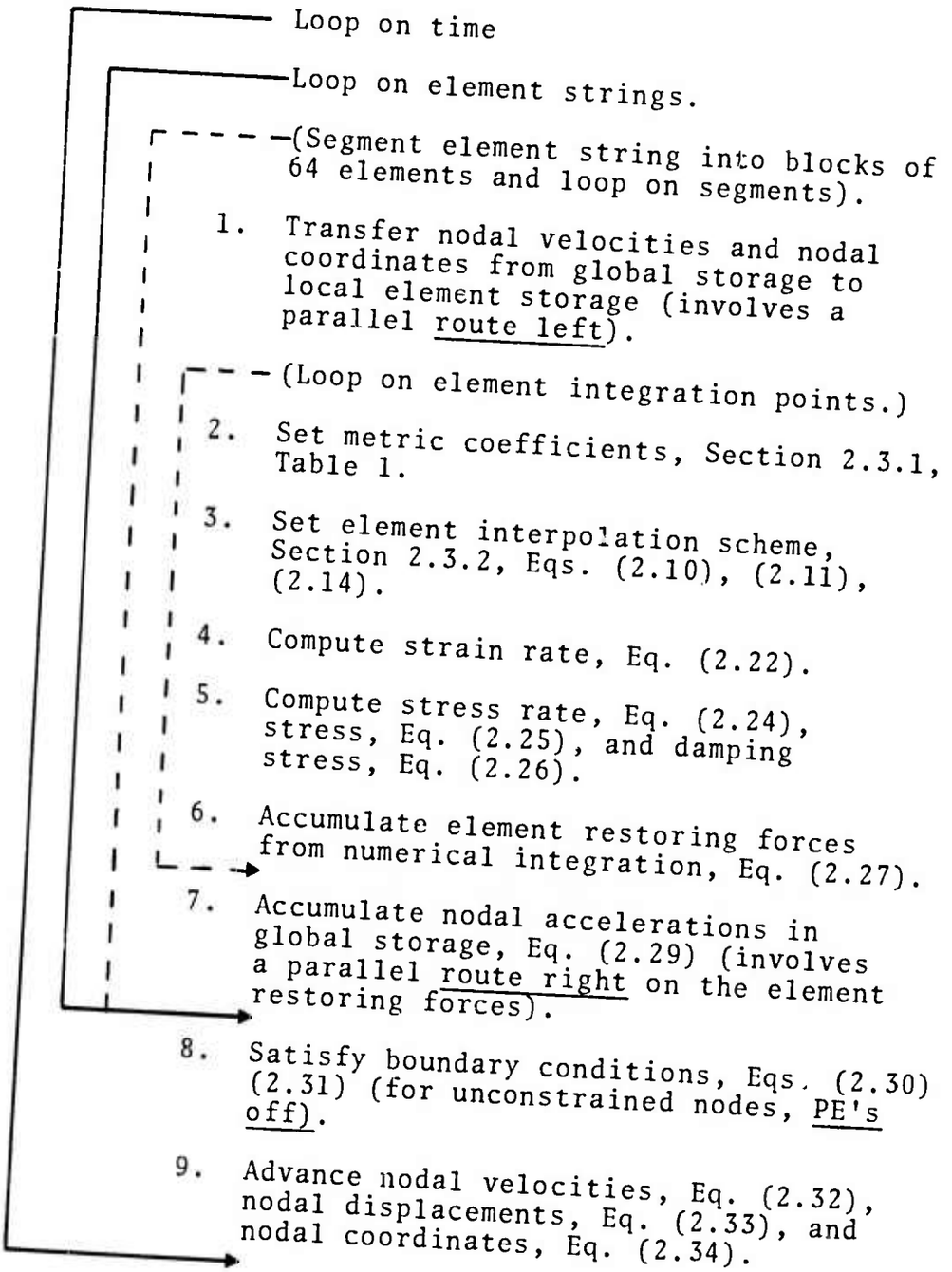


Fig. 3.5--Parallel computing scheme used in the SWIS code.

Strings of 64 elements are sequentially processed with a parallel route left (global to element storage) in the initial operation and a parallel route right (element to global storage) in the final operation.

The acceleration of each node in the grid is determined from the processing of all element strings. Boundary conditions are then applied by sweeping through global storage, modifying accelerations of only those nodes with force, displacement, or fixed boundary conditions. At the same time, the advanced nodal velocities and displacements are computed in parallel by time integration. This operation completes one numerical time step.

3.3.5 Code Output

For the testing stage of SWIS, we have relied on two basic types of numerical output. Both of these employ the DISPLAY software facility for outputting formatted data during run time.

One method is to output an entire PE row of information at a selected time step. Since an interesting portion of the problem often has its orientation across PE's, row output can provide the desired results in condensed form. One test problem – Lamb's problem as described in the next chapter – had the free surface of the grid oriented along the first grid dimension. Thus six row outputs provided horizontal and vertical components of displacement velocity and acceleration along the entire free surface. This approach is less convenient for grid cross-sections other than those oriented across PE boundaries. A mechanism is used for transferring a series of selected quantities into a buffer row of memory for output.

A similar mechanism is used for collecting time histories. The displacement, or some other quantity at selected points in the grid, is accumulated into a buffer area following each time step. When the buffer is full, it is output and the accumulation is resumed. In this way, the behavior of a point in the grid over time may be conveniently displayed.

We anticipate incorporating further modes of program output into SWIS. One is a binary output mode in which large quantities of unformatted data are transferred to UNICON laser storage during run time. Selected portions of this file could be transferred over the ARPANET for printing or plotting at another site. A further possibility is to incorporate some code into SWIS for generating plots using the network graphics protocol (NIC #15358, 1973). This plot information could be plotted at any site equipped with preprocessors for the graphics protocol.

IV. STRESS WAVE CALCULATIONS

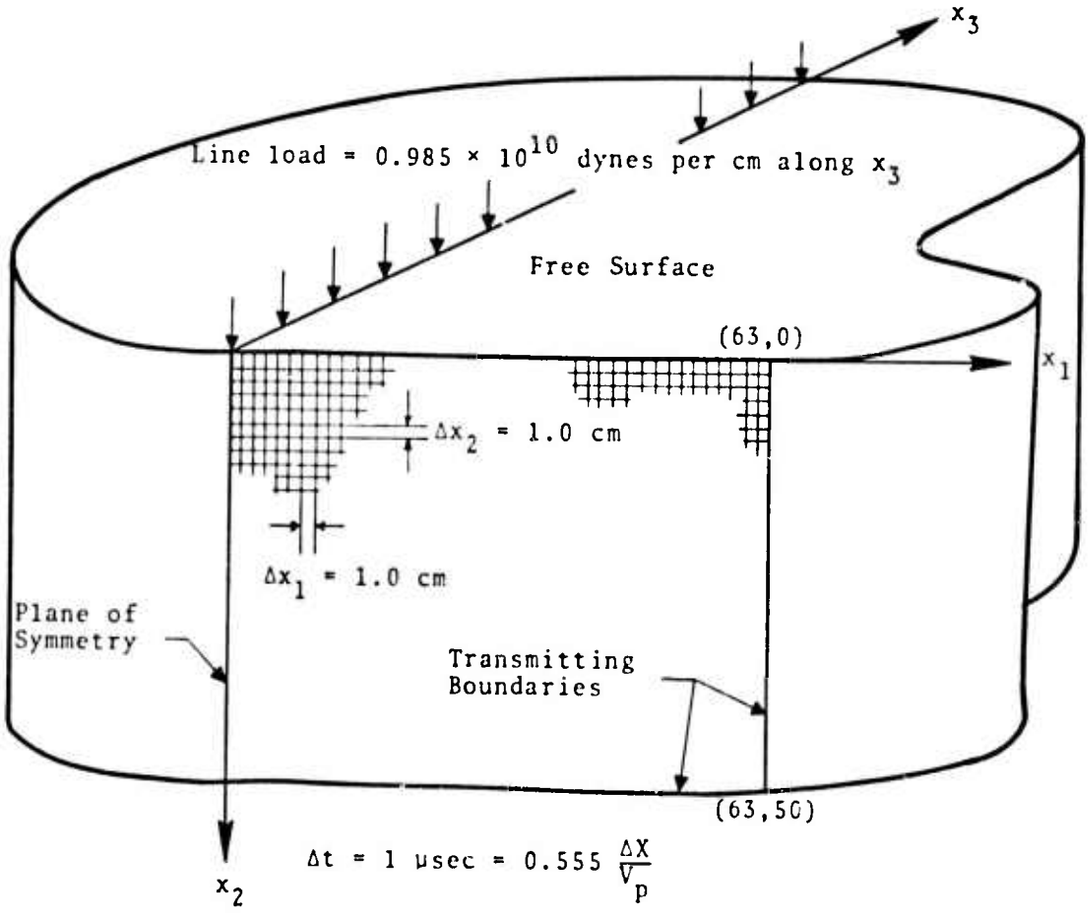
Our primary thrust during this contract period has been to develop code on the ILLIAC for performing 3-D stress wave calculations. Test calculations have been performed in 1-D, 2-D, and 3-D geometries to verify the resulting code and to examine particular features of the computing algorithm. A sample of these test calculations are presented below.

4.1 LAMB'S PROBLEM: 2-D CARTESIAN COORDINATES

The ILLIAC SWIS code has been exercised in 2-D Cartesian geometry (plane-strain) for treating Lamb's problem: A line load (a point load in the plane of the grid) applied as an impulse to the free surface of a half-space. Figure 4.1 illustrates the numerical presentation and provides the physical parameters that were used in the calculation. Based on central system clock times, we estimate that the calculations were processed at the rate of 0.4 m-sec per 2-D element per numerical time step.

The sharp wave forms that arise from the concentrated impulse loading serve as a critical test for the stress wave computing scheme. Frazier, et al. (1973) have reported on finite element and finite difference treatments of Lamb's problem using S^3's UNIVAC 1108. In the present treatment of Lamb's problem on the ILLIAC, we have examined alternate schemes for dealing with bending modes in the individual elements (hour glass deformations, see Appendix A) and alternate schemes for damping spurious high frequency noise. We have also made an effort to test the effectiveness of transmitting boundary conditions in 2-D geometry.

Computed displacements along the free surface are compared with mathematical solutions in Figs. 4.2, 4.4 – 4.6. In the first of this series of calculations, Figs. 4.2 and



Physical Parameters

ρ = mass density = 2.77 g/cm^3

V_p = P wave velocity = 0.555 cm/sec

V_s = S wave velocity = 0.3145 cm/μsec

V_r = Rayleigh wave velocity = 0.2898 cm/μsec

L = total impulse = $\iint$ pressure $dx_1\ dt$

= 1.97×10^{10} dyne-μsec/cm

Fig. 4.1--Problem description used for analyzing a line load impulse on a half-space (Lamb's problem).

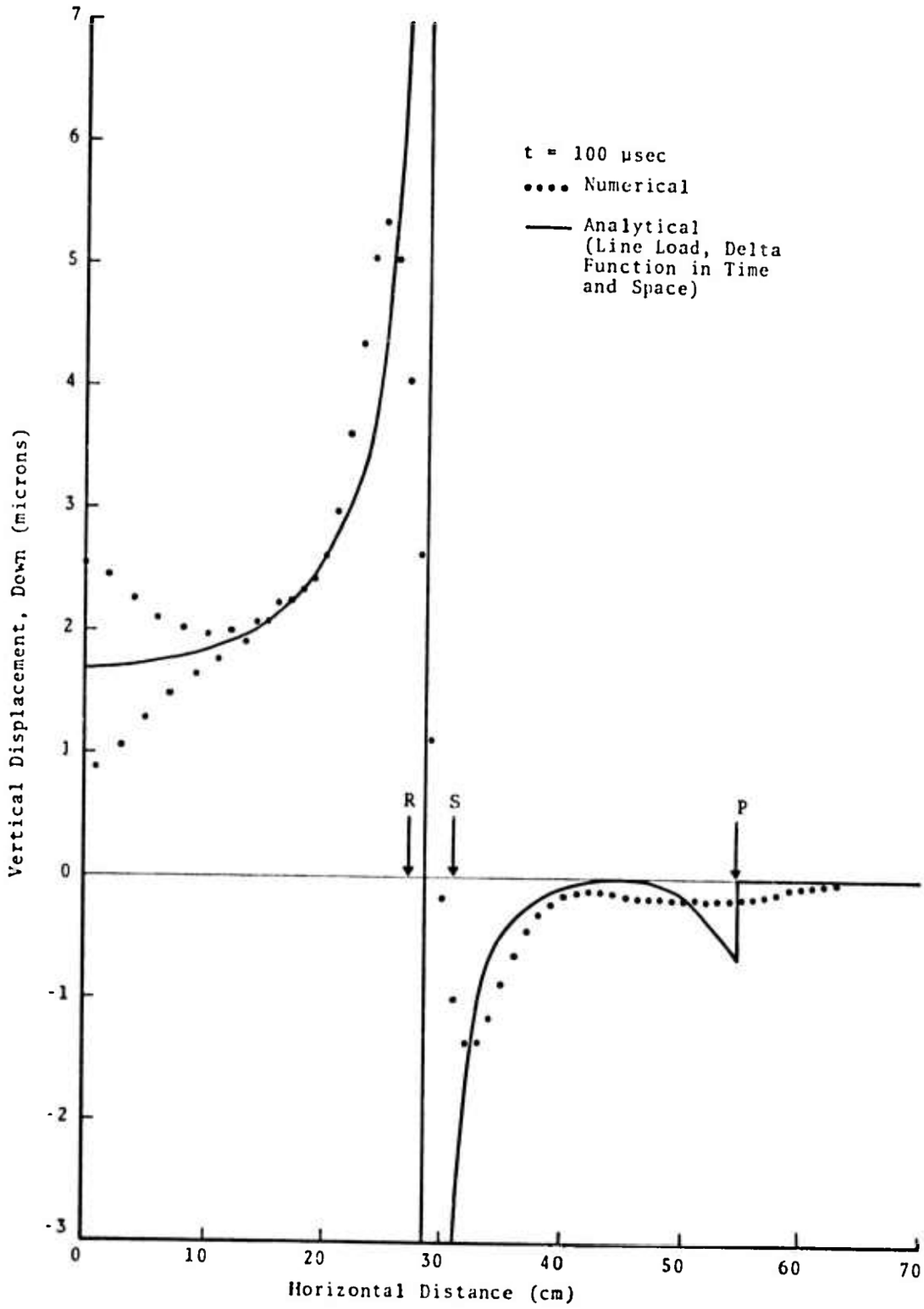


Fig. 4.2--Vertical displacement of the free surface: Damping coefficient β_P = 0.80 for P waves, β_S = 0.80 for S waves, and β_H = 0.80 for hour glass motions. Inner element stress variations due to bending are not included.

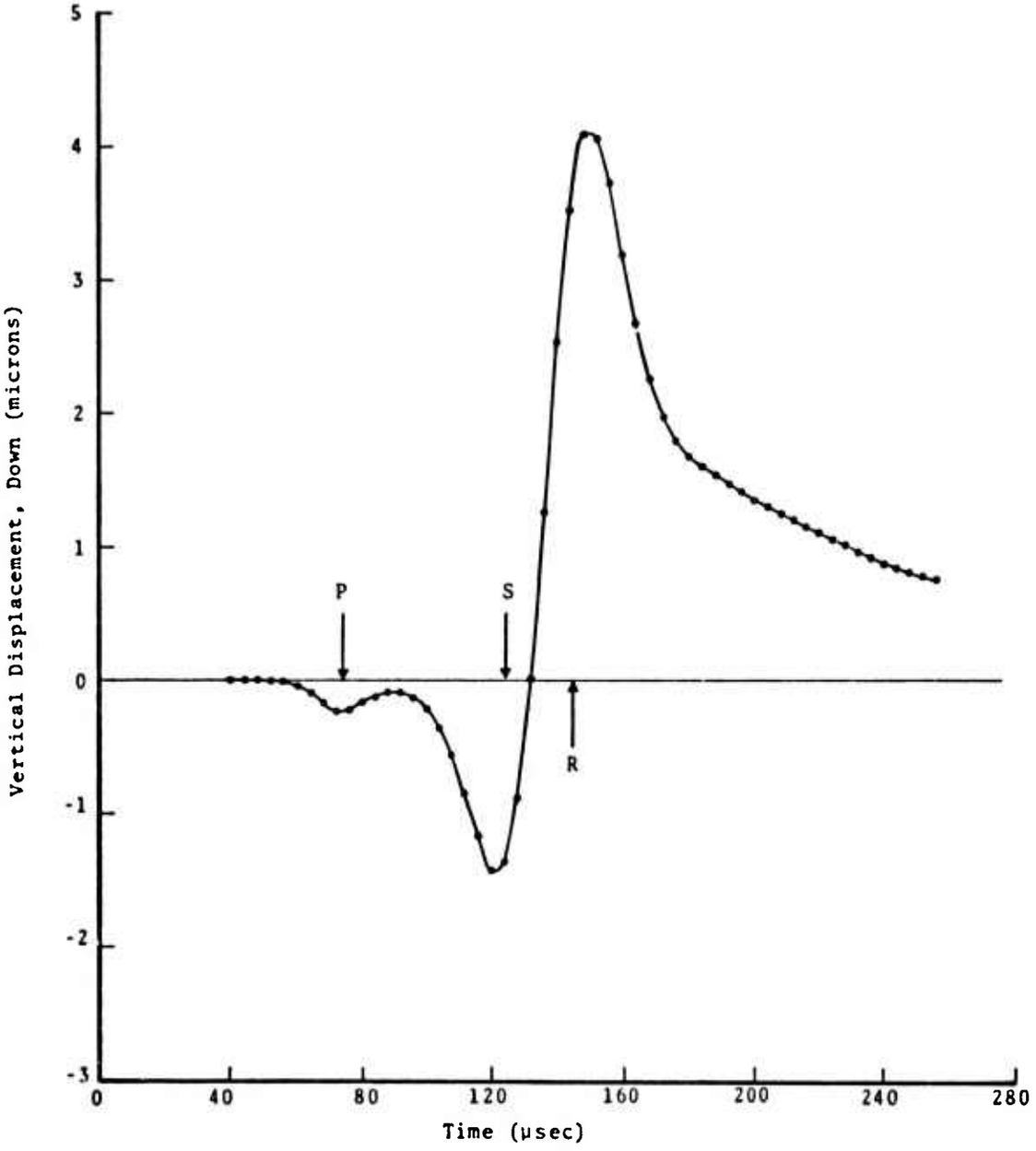


Fig. 4.3--Time history of a point on the free surface, 39 cm from the applied load. Damping coefficient $\beta_P = 0.80$ for P waves, $\beta_S = 0.80$ for S waves, and $\beta_H = 0.80$ for hour glass motions. Inner element stress variations due to bending are not included.

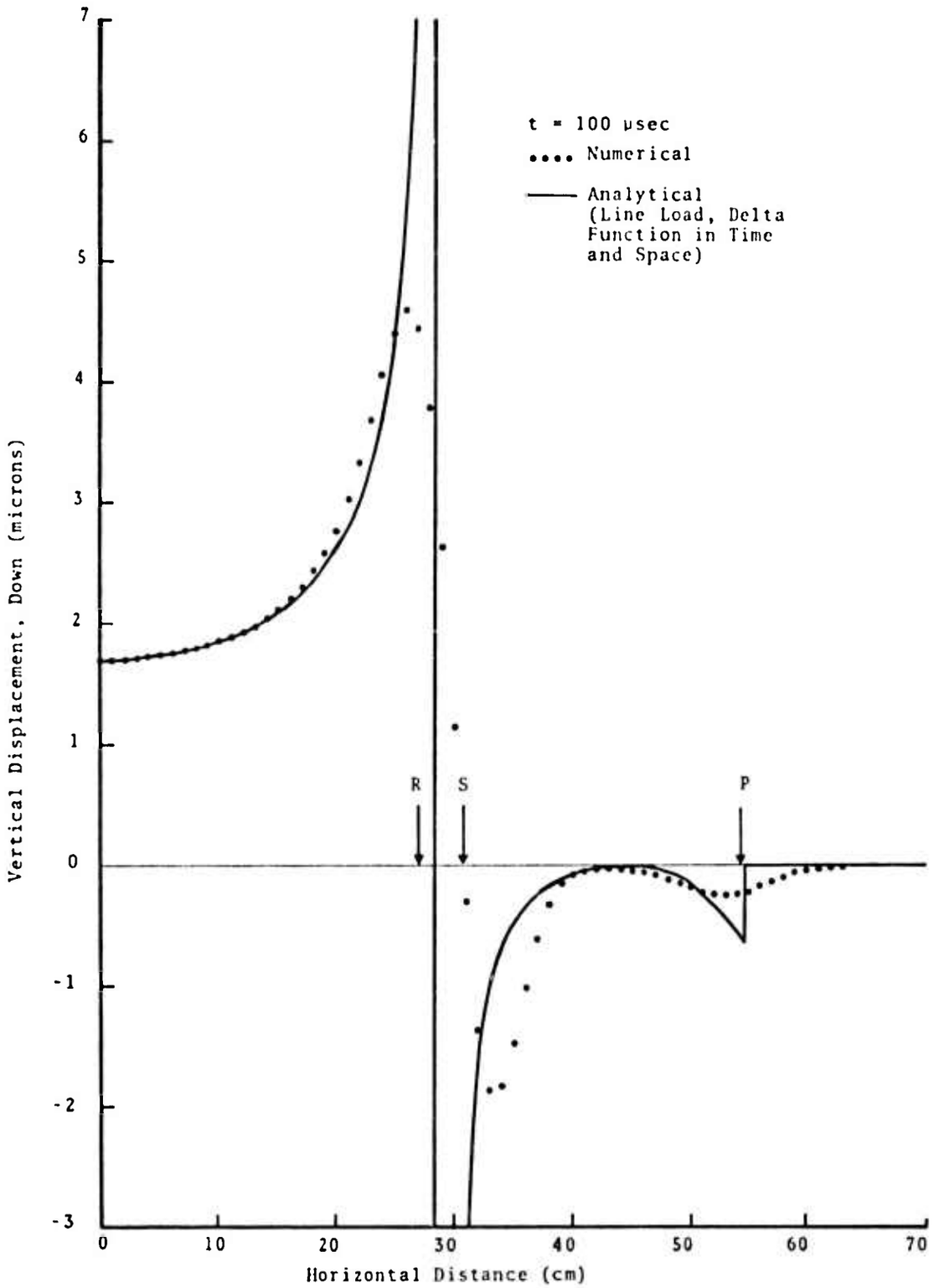


Fig. 4.4--Vertical displacement of the free surface: Damping coefficient β_P = 0.40 for P waves, β_S = 0.80 for S waves, and β_H = 0.40 for hour glass motions. Inner element stress variations due to bending are included.

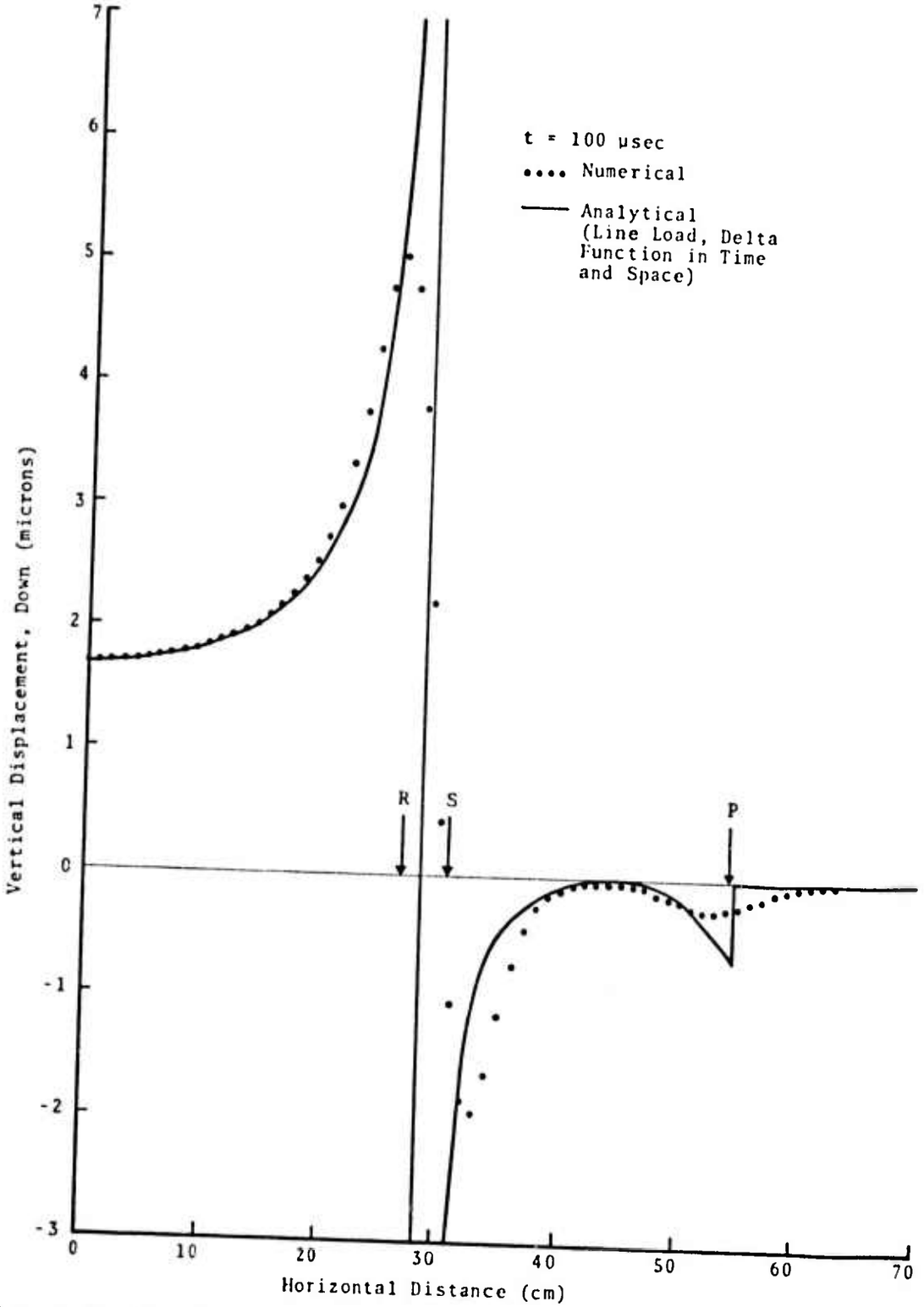


Fig. 4.5--Vertical displacement of the free surface: Damping coefficient β_P = 0.40 for P waves, β_S = 0.80 for S waves, and β_H = 0.40 for hour glass motions. Fifty percent of inner element stress variations due to bending are included.

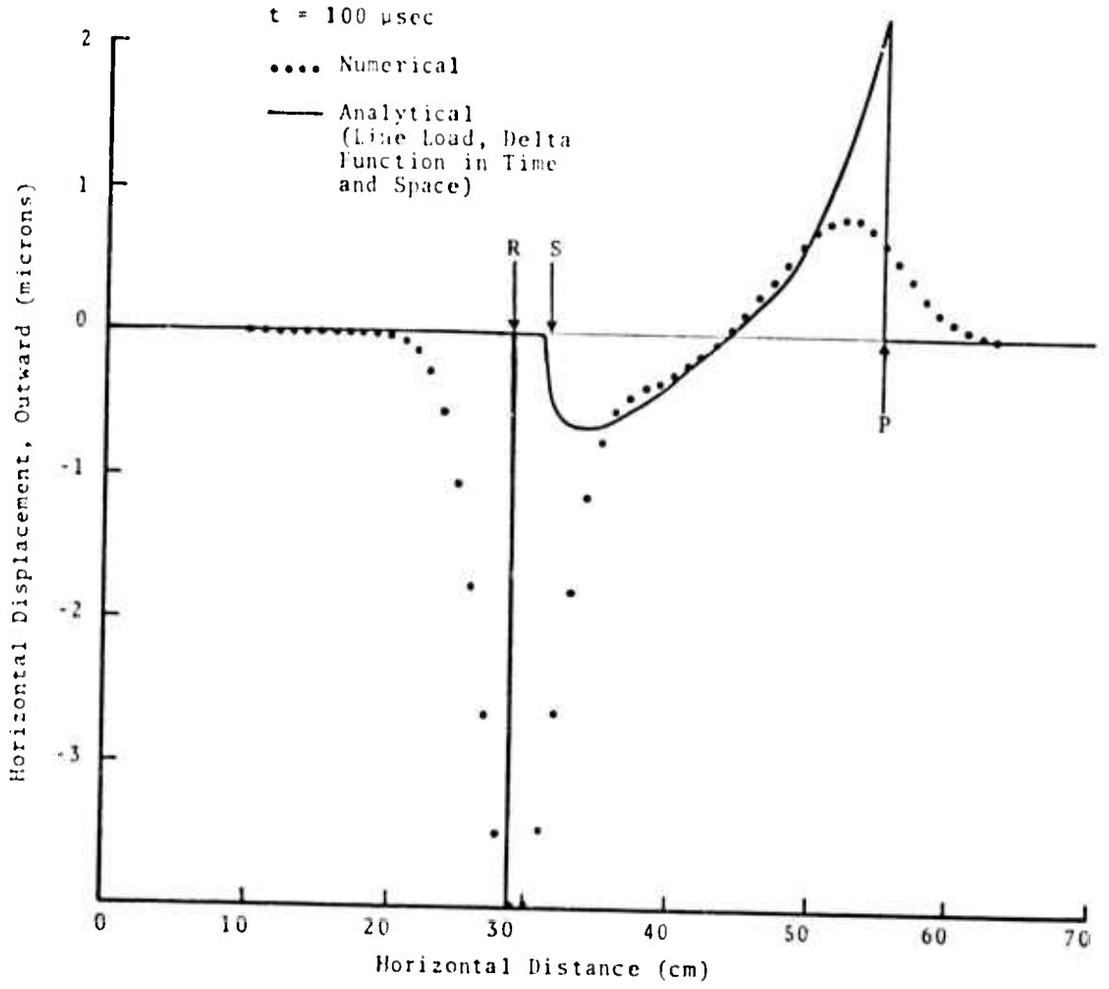


Fig. 4.6--Horizontal displacement of the free surface: Damping coefficient $\beta_P = 0.40$ for P waves, $\beta_S = 0.80$ for S waves, and $\beta_H = 0.40$ for hour glass motion. Fifty percent of inner element stress variations due to bending are included.

4.3, a single-point integration scheme was employed for computing restoring forces of the medium (see Eq. (2.27) or alternately Eq. (2.46)). As discussed in Appendix A, this scheme does not take into account the spatial variations in stress within an element. Consequently, the elements contain no stiffness against bending. The result is that the concentrated load configuration excites hour glass deformation modes in the grid in addition to the desired wave forms. An effort was made to control this undesirable response by introducing a special artificial viscosity with the sole purpose of damping the hour glass deformations. While this procedure was successful in suppressing the hour glass modes from the computed velocities, we see from Fig. 4.2 that considerable numerical noise appears in the displacement field in the vicinity of the applied load. By employing the hour glass damping scheme, a well-formed Rayleigh wave emerges at points greater than about 20 grid dimensions from the source. The time history of a point 39 cm from the source, presented in Fig. 4.3, shows no trace of hour glass noise. We note that no special treatment is required to control hour glass deformation modes when the surface load is distributed over many elements; there is essentially no excitation of hour glass modes in this case.

The SWIS code was then altered to take account of inner element variations in stress, rather than to artificially damp the hour glass modes. The vertical component of displacement along the free surface, using this numerically consistent formulation, is presented in Fig. 4.4. Also, the standard numerical damping scheme, Eq. (2.26), was altered somewhat for this calculation. Shear distortions were damped using a coefficient $\beta_S = 0.80$, the same β that was used in the previous calculation; whereas a smaller damping coefficient of 0.40 was applied to

volumetric distortions. This procedure has the effect of damping high frequency P and S waves of comparable wave lengths rather than damping comparable frequencies as is the case using one β for both volumetric and distortional deformations (see Frazier, et al., Appendix B, 1973).

First, we see from the results presented in Fig. 4.4 that there is no trace of hour glass deformation present in the computed displacement field. Also, we find that the reduction in the artificial damping of volumetric distortions results in a more distinct P wave along the surface.

As a final numerical experiment, the restoring forces that arise from bending of individual elements were artificially reduced by 50 percent. This inconsistent analysis was performed to examine what effect the bending stiffness of individual elements has on the resulting wave forms. Cell-centered-stress finite difference schemes disregard the bending stiffness of individual elements; while the consistent finite element scheme, used to produce Fig. 4.4, slightly over-estimates the bending stiffness of elements. We see from Fig. 4.5 that a slightly sharper Rayleigh wave (higher peak displacement) is obtained when the bending stiffness of the elements is reduced. It appears that an accurate representation for the bending stiffness of individual elements is not critical to this particular wave simulation. Except for the hour glass deformations that permeate the displacement field near the point of loading (Fig. 4.2), it appears that artificial damping, which is needed to control numerical dispersion, has as much effect on peaked wave forms as the bending stiffness of the elements.

Finally, some data on machine reliability have been obtained from the numerical simulations of Lamb's problem. Eight separate calculations were initiated, each set for

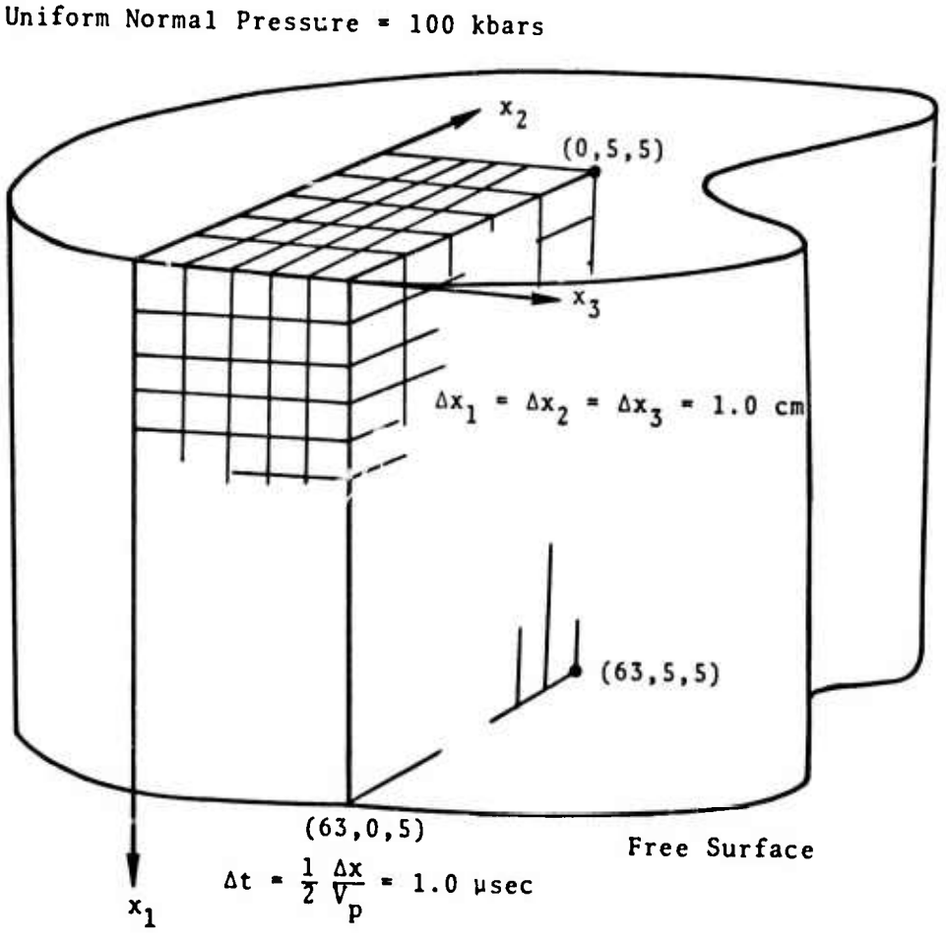
500 time steps. Major errors, which are detected by orders of magnitude increase in the total energy, occurred in all eight calculations prior to completion. Two calculations did not reach 100 time steps before the total energy jumped; one calculation progressed just beyond 250 time steps before major spurious errors occurred. Each numerical time step requires about 3×10^5 floating point multiply and add operations, approximately 100 operations per element per time step. Thus, it appears that bits are being altered in the exponents at the rate of about one error per 3×10^7 floating point multiply and add operations for the particular sequence of machine instructions activated by this 2-D wave simulation. In some instances, we also find that low order errors have occurred in the calculations before errors of very large magnitudes appear. We plan to incorporate rigorous energy checks and an automatic restart mechanism in the SWIS code to detect and recover from machine errors.

4.2 PLANE WAVES: 3-D CARTESIAN COORDINATES

The first test calculation of the SWIS code in 3-D geometry has been designed to simulate a plane wave produced by an impulse pressure. As with the problem of the previous section, the propagating wave has a discontinuity in the displacement field and a singularity in the velocity field. We do not expect to match behavior of this type accurately but rather to examine the limitations of the numerical procedure. The response of a linear system to an impulse load provides the Green's function for the system. Thus, by treating an impulse loading, we obtain the Green's function for the numerically discrete system.

The grid configuration, grid parameters, and material parameters are presented in Fig. 4.7. The grid, which is core-contained, involves 1575 cubic elements with 63 grid spaces in the direction of propagation. The uniform pressure pulse produces particle motion only in the direction of loading. Computed displacement and velocity at selected time intervals (t = 40, 80, 120 μsec) are presented in Fig. 4.8. The computed velocity field represents a numerical approximation to a delta function. The time integral of the computed particle velocity matches the time integral of the analytical singularity. This is illustrated by the close agreement between computed and analytical displacement fields behind the wave front.

Based on central system clock times, we estimate that the 3-D plane wave calculations were processed at the rate of 1.2 m-sec per element per numerical time step.



Physical Parameters

ρ = mass density = 2.5 g/cm^3

V_p = P wave velocity = 0.5 cm/μsec

V_s = S wave velocity = 0.25 cm/μsec

β = damping coefficient = 0.15

Fig. 4.7--Problem description used to simulate a uniform impulse pressure applied to the surface of a half-space.

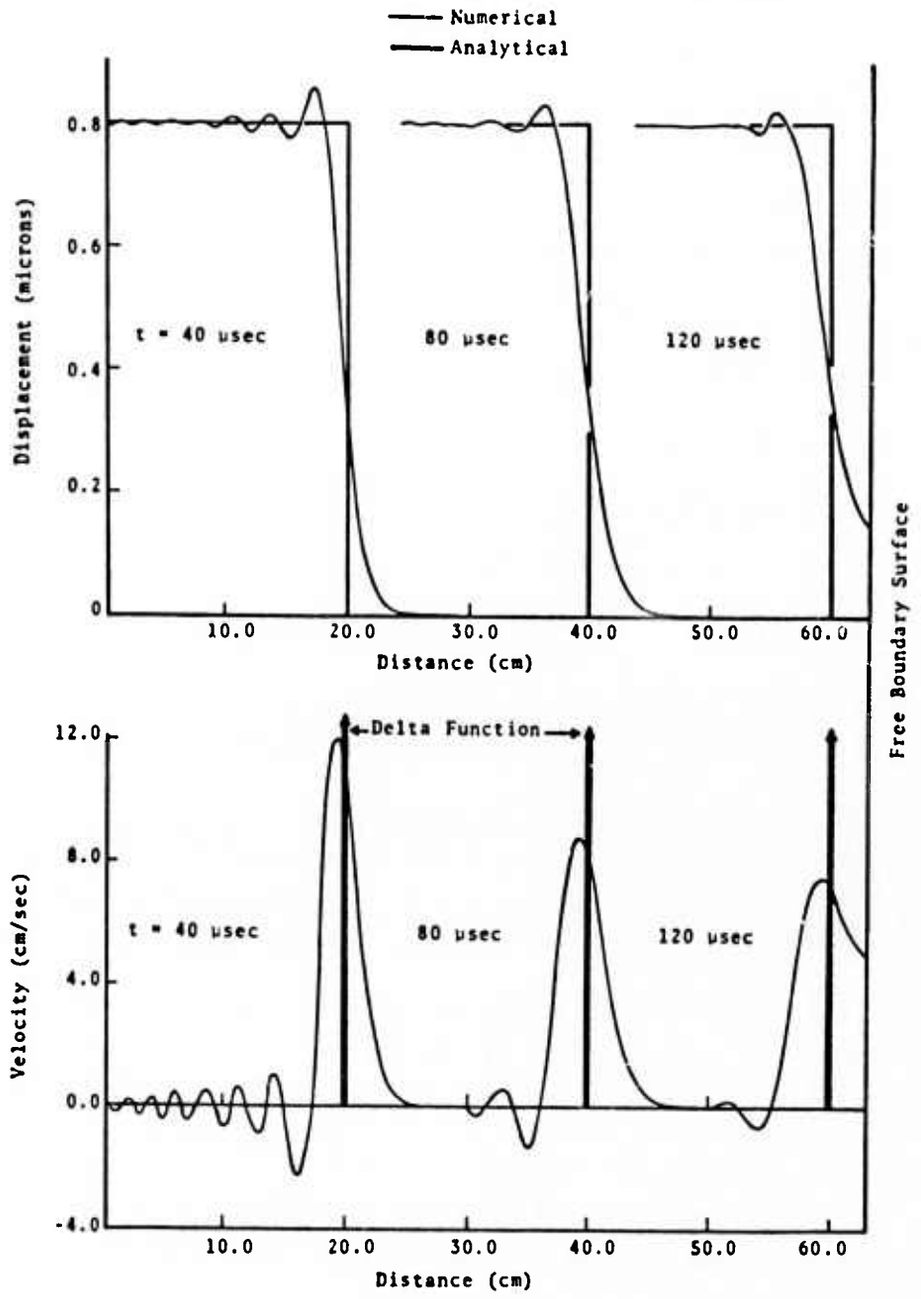


Fig. 4.8--Plane wave produced by a uniform impulse pressure.

4.3 BURIED EXPLOSION: 1-D, 2-D, 3-D SPHERICAL COORDINATES

The facility in the SWIS code for processing wave calculations in curvilinear coordinates is employed for simulating elastic compression waves emanating from a spherical cavity. A step pressure with an exponential decay is analyzed using the following parameters:

r_c = cavity radius = 10 m

Δr = radial zone size = 0.2 m

Δt = time step = 20 μsec

β = damping coefficient - 0.15

V_p = P wave velocity = 5 km/sec

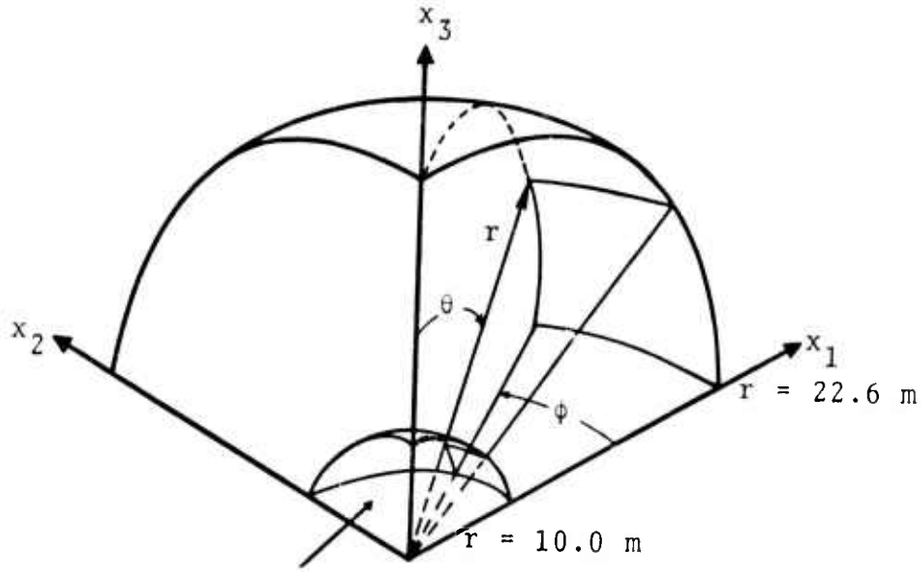
V_S = S wave velocity = 2.5 km/sec

ρ = mass density = 2.0 g/cm^3

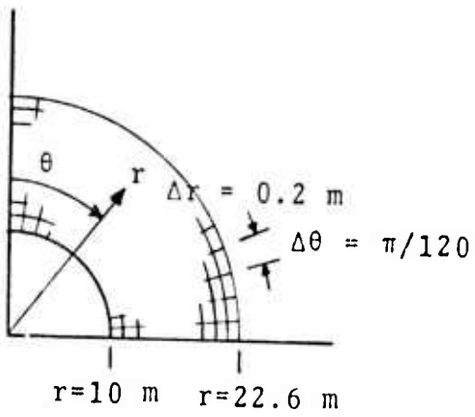
$p(t)$ = cavity pressure = $e^{-10^3 t}$ kbar

The elastic explosion calculation is processed in spherical coordinates using 1-D, 2-D, and 3-D grid configurations, illustrated in Fig. 4.9. Each grid has 63 elements in the radial dimension. This provides equal resolution for each grid configuration in the spherically symmetric explosion. Consequently, we would expect very nearly identical results from the 1-D, 2-D, and 3-D calculations.

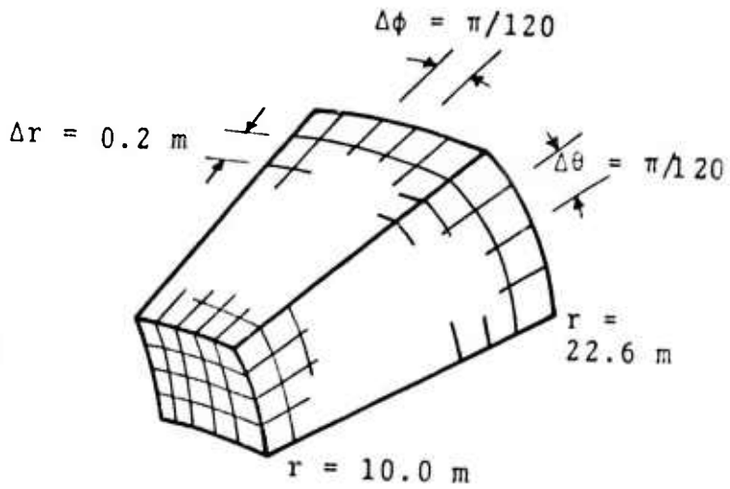
The computed response of the cavity wall is compared with analytic solutions by Blake (1952) in Fig. 4.10. Radial displacement and velocity are compared 2.0 m-sec after pressurizing the cavity with the analytic solutions in Fig. 4.11. While the computed points in Figs. 4.10 and 4.11 are taken from the 1-D spherical calculation, the 2-D and 3-D calculations gave results with 1 percent of the 1-D calculation. Therefore, these figures can be considered representative of the computed behavior for all



Pressurized Cavity



2-D Grid



3-D Grid

Fig. 4.9--Grid configurations for analyzing pressurized spherical cavity: 1-D, 63 Δr; 2-D, 63 Δr by 60 Δθ; 3-D, 63 Δr by 5 Δθ by 5 Δφ.

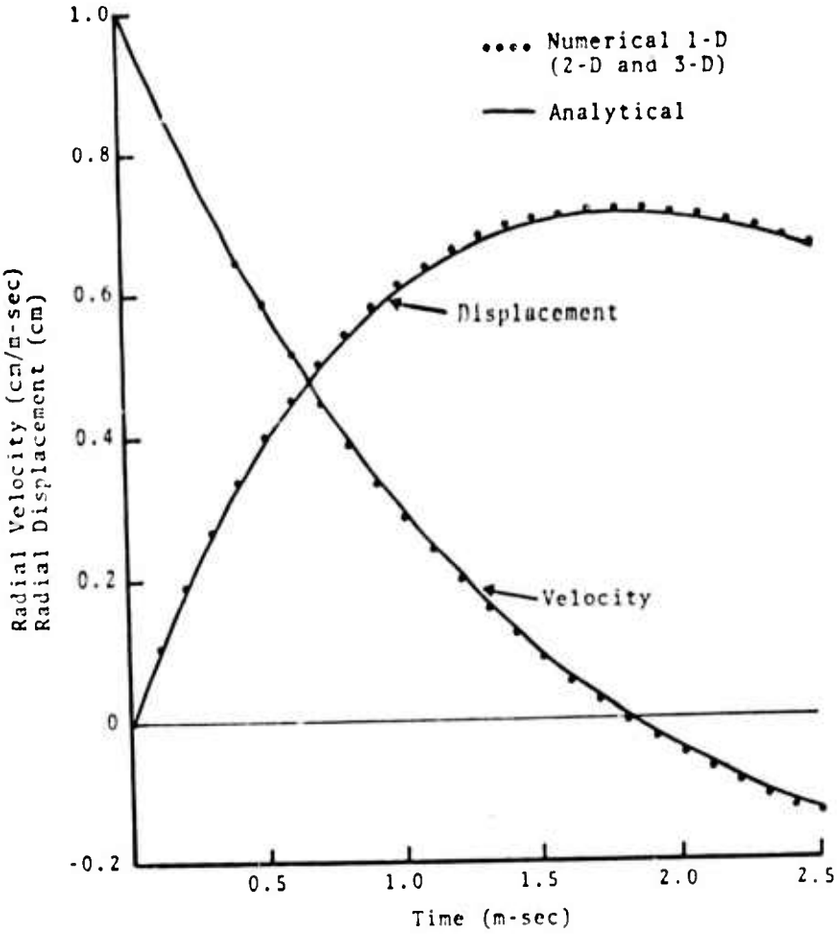


Fig. 4.10--Radial velocity and displacement of cavity wall; 2-D and 3-D results within 1 percent of plotted points for 1-D calculation.

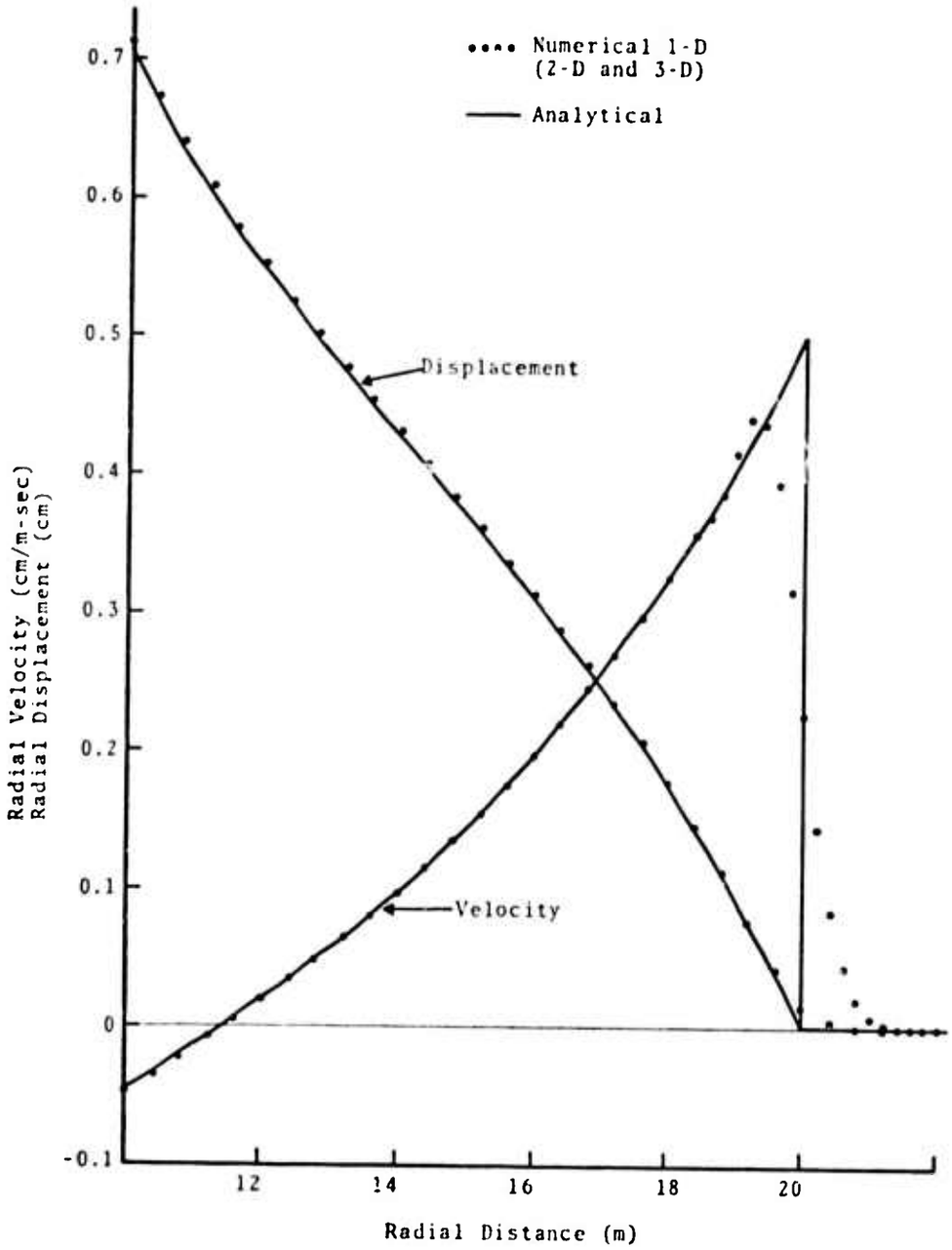


Fig. 4.11--Radial velocity and displacement 2.0 m-sec after pressurizing spherical cavity; 2-D and 3-D results within 1 percent of plotted points for 1-D calculation.

three grid configurations.

More test calculations need to be performed to verify the SWIS code for other curvilinear coordinate systems. However, because no code alterations were needed to produce the 2-D and 3-D spherical results, it appears likely that the curvilinear formulation is generally operational. We note that the wave calculations processed in spherical coordinates required about 50 percent more computer time than would have been required for a comparable Cartesian calculation.

V. SUMMARY AND CONCLUSIONS

Considerable progress has been made in the development of an ILLIAC computer code for simulating stress waves in the earth. A novel finite element scheme has been developed for processing stress waves in 1-D, 2-D, and 3-D curvilinear geometries. The numerical scheme accommodates finite deformations and nonlinear material behavior using a Lagrangian formulation.

In the initial phase of this program, code development was confined to small amplitude stress waves with linear material response. During this period, an ILLIAC code was developed for time stepping stress waves through highly irregular 1-D, 2-D, and 3-D grid configurations. The linear code was made operational on the ILLIAC, and test calculations were performed as early as March 1973 to verify the code. The linear code development effort on the ILLIAC proved valuable in the subsequent development of the more general SWIS code.

Because of the manner in which the more recent nonlinear SWIS code has been adapted to the parallel structure of the ILLIAC, grid points are associated with PE's. This has required a certain regularity in the grid configurations, namely, the grid must be composed of an assemblage of element strings. The parallel processing capabilities of the ILLIAC are most effectively utilized when the number of elements in the element strings is some multiple of 64. This is in contrast with the earlier linear code which processes irregular grids just as efficiently as regular grids.

The nonlinear SWIS code was programmed using GLYPNIR and debugged directly on the ILLIAC. Successful test calculations were performed on the ILLIAC just three months

after the initial programming was begun. This fact is a credit to the total ILLIAC system: the ARPANET, the ILLIAC IV computer, and the peripheral equipment at the ILLIAC site. Systems failures have interfered with our ability to utilize the ILLIAC on the average of about two weeks out of each month for the past three months. System reliability is expected to improve in the coming months.

A series of core-contained test calculations have been performed on the ILLIAC to verify the SWIS code in 1-D, 2-D, and 3-D Cartesian and spherical coordinates. Cartesian coordinates were employed to simulate a line load impulse (Lamb's problem) using 3150 2-D elements and a pressure pulse using 1575 3-D elements. Spherical coordinates were employed to simulate a pressurized spherical cavity using 63 1-D elements, 3780 2-D elements, and 1575 3-D elements. The calculations were processed at the rate of 0.4 and 1.2 m-sec per element per numerical time step for the 2-D and 3-D Cartesian grids, respectively. Approximately 50 percent slower computing rates were obtained for the spherical calculations. Based on repetitive executions of Lamb's problem, we have observed what appears to be machine errors. We estimate that errors occur in the exponent bits at the rate of one error per 3×10^7 floating point multiply and add operations. Errors in the mantissa, which are slightly more difficult to detect, have also been observed. We plan to incorporate rigorous energy checks and an automatic restart mechanism in the SWIS code to detect and recover from machine errors.

REFERENCES

Blake, F. G., "Spherical Wave Propagation in Solid Media," Jour. Acoust. Soc. Am., Vol. 24, pp. 211-215, 1952.

Frazier, G. A., J. H. Alexander, and C. M. Petersen, "3-D Seismic Code for ILLIAC IV," Systems, Science and Software Report SSS-R-73-1506, Contract No. DNA 001-72-C-0154, February 9, 1973.

Network Graphics Protocol, NIC #15358, Stanford Research Institute, April, 1973.

Richtmyer, R. D., and K. W. Morton, Difference Methods for Initial-Value Problems, Interscience Publishers, Second Edition, 1967.

Spain, B., Tensor Calculus, Interscience Publishers, Third Edition, 1960.

Washizu, K., Varational Methods in Elasticity and Plasticity, Pergamon Press, 1968.

APPENDIX A

INNER-ELEMENT STRESS VARIATIONS

The stress tensor evaluated at the centroid of a 2-D or 3-D rectilinear element does not adequately describe the state of stress in the element. We note that no stress is generated at the centroid as the element undergoes a bending deformation, illustrated in Fig. A.1. Stress wave calculations that use centroidal stresses exclusively can result in hour-glass deformation patterns superposed on the computed displacements.

Let A denote the amplitude of the bending mode pictured, then

$$A = < +1, -1, -1, +1 > \{u_1^e\}$$

$$u_1(\underset{\sim}{z}) = A z_1 z_2$$

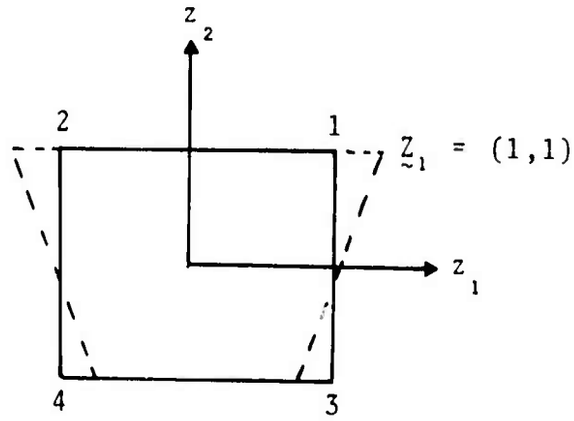
$$\varepsilon_{ij}^e(\underset{\sim}{z}) = \frac{1}{2}\left(\frac{\partial u_i}{\partial z_j} + \frac{\partial u_j}{\partial z_i}\right) = A \begin{bmatrix} z_2 & \frac{1}{2} z_1 \\ \frac{1}{2} z_1 & 0 \end{bmatrix}$$

$$\sigma_{ij}^e(\underset{\sim}{z}) = 2\mu\varepsilon_{ij} + \delta_{ij}\lambda\varepsilon_{kk} = A \begin{bmatrix} (\lambda+2\mu) z_2 & \mu z_1 \\ \mu z_1 & \lambda z_2 \end{bmatrix}$$

$$\sigma_{ij}^e(\underset{\sim}{z} = 0) = 0$$

on the desired displacement and velocity fields.

A one-point integration scheme is not sufficient for treating spatial variations in stress with an element. When two integration points per element dimension are used in Eq. (2.20) for computing restoring forces of the medium, considerably more calculations are required to process each



Let A denote the amplitude of the bending mode pictured, then

$$A = \langle +1, -1, -1, +1 \rangle \left| U_1^e \right|$$

$$u_1(\underset{\sim}{z}) = A\, z_1 z_2$$

$$\epsilon_{ij}^e(\underset{\sim}{z}) = \frac{1}{2}\left(\frac{\partial u_i}{\partial z_j} + \frac{\partial u_j}{\partial z_i}\right) = A\begin{bmatrix} z_2 & \frac{1}{2} z_1 \\ \frac{1}{2} z_1 & 0 \end{bmatrix}$$

$$\sigma_{ij}^e(\underset{\sim}{z}) = 2\mu\epsilon_{ij} + \delta_{ij}\lambda\epsilon_{kk} = A\begin{bmatrix} (\lambda+2\mu) z_2 & \mu z_1 \\ \mu z_1 & \lambda z_2 \end{bmatrix}$$

$$\sigma_{ij}^e(\underset{\sim}{z}=0) = 0$$

Fig. A.1--Bending mode of deformation with nonuniform stress tensor that vanishes at the centroid.

element. However, a two-point integration scheme is sufficiently accurate to treat quadratic variations in stress; consequently, little additional computing effort would be required to process higher order elements.

For the increased accuracy of a two-point integration scheme to be effective, a more complete description of the spatially varying element stresses is required. We note two possibilities: (1) Store element stress at each integration point. This procedure requires a considerable amount of storage. (2) Store the restoring forces at node points and update the restoring forces using stress rates evaluated at each integration point. This procedure is currently being installed in the SWIS code.

Alternate procedures were used for processing Lamb's problem, presented in Section IV. We chose to treat a point load (applied at one discrete point in time and space) as a critical test for the numerical computing scheme in preference to a distributed load for which the hour-glass mode is negligible. As a first effort, we simply damped the bending deformations in each element. This procedure succeeded in removing hour-glass patterns from the velocity field, however, hour-glass deformations appeared in the displacement field in the vicinity of the impulsively loaded surface. Lamb's problem was then repeated by including the restoring terms that arise from bending deformations in the element. As expected, no hour-glass patterns appear in these results.

APPENDIX B

The following scheme for banded sparse matrix multiplies was developed by Frazier (1972). The sparse matrix involved is a matrix of influence coefficients $[\underset{\approx}{A}]$, which is a $N \times N$ matrix of 3×3 submatrices. As noted in Section 3.2, the uncompressed matrix requires roughly 10^9 words of storage for a 3-D problem containing $N = 10^4$ nodes. However, since most nodes have just 26 immediate neighbors in a 3-D rectilinear gridwork, there will usually be 27 non-zero submatrices in a single row of

$$[\underset{\approx}{A}] = \underset{\approx}{A}_{n,m}; \; n,m = 1,2, \ldots N \; .$$

The unnecessary zeroes are compressed out of $[\underset{\approx}{A}]$ to yield a $N \times 27$ matrix of 3×3 submatrices. From the node numbering sequence we can deduce the column numbers for the non-zero terms in each row, i.e.,

$$m = m_{n,k} \; , \; n = 1,2, \ldots N$$
$$k = 1,2, \ldots \approx 27 \qquad \text{(B.1)}$$

where n and m are the row and column numbers of $[\underset{\approx}{A}]$, respectively, and N is the total number of node points in the 3-D grid. The array of contributing column numbers $m_{n,k}$, $k = 1,2, \ldots \approx 27$ are simply the node numbers adjacent to node n.

Only the non-zero multiplications of the matrix multiply

$$\underset{\sim}{B}_n = \sum_{m=1}^{N} \underset{\approx}{A}_{nm} \underset{\sim}{U}_m \; , \; n = 1,2, \ldots N \qquad \text{(B.2)}$$

are performed in the sparse matrix multiply, which can now

be expressed as

$$\underset{\sim}{B}_n = \sum_{k=1}^{\approx 27} \underset{\approx}{\overline{A}}_{n,k} \, \underset{\sim}{U}_{m_{n,k}} \tag{B.3}$$

for $n = 1,2, \ldots N$ with

$$\underset{\approx}{\overline{A}}_{n,k} = \underset{\approx}{A}_{n,m_{n,k}}$$

The compressed matrix $[\underset{\approx}{\overline{A}}]$ should be arranged on the disk so that each term arrives in the processor containing the nodal displacement for which it is to be multiplied, Eq. (B.3). Thus, $\underset{\approx}{\overline{A}}_{n,k}$ should arrive in the PEM containing $\underset{\sim}{U}_{m_{n,k}}$ without requiring additional shift operations. If $[\underset{\approx}{\overline{A}}]$ remains unchanged over many multiply operations, considerable effort can be devoted to arranging the non-zero terms of the matrix on mass storage in an optimal fashion for processing.

In the case of the 3-D grid, the vectors $\underset{\sim}{U}_m$ are nodal displacements of three components and require three numbers for their representation. Let U_{im} be the component of $\underset{\sim}{U}_m$ along the x_i axis, $i = 1,2,3$. Correspondingly, $\underset{\approx}{\overline{A}}_{n,k}$ is a 3×3 matrix that requires nine numbers in its representation. Let $\overline{A}_{in,jk}$ denote the ij element of this matrix. Then Eq. (B.3) can be written

$$B_{in} = \sum_{k=1}^{\approx 27} \sum_{j=1}^{3} \overline{A}_{in,jk} \, U_{jm_{n,k}} \tag{B.4}$$

where B_{in} is the component of $\underset{\sim}{B}_n$ along the x_i axis.

The nodal displacements are arranged on the disk to flow into the PEM's (denoted p) by PE rows (denoted

r) so that $\underset{\sim}{U}_{11} \rightarrow p = 0, r = 0$; $\underset{\sim}{U}_{21} \rightarrow p = 0, r = 1$; $\underset{\sim}{U}_{31} \rightarrow p = 0, r = 2$; $\underset{\sim}{U}_{12} \rightarrow p = 1, r = 0$; ... $\underset{\sim}{U}_{3,64} \rightarrow p = 63, r = 2$; $\underset{\sim}{U}_{1,65} \rightarrow p = 0, r = 3$; etc.

In general we have

$$U_{jm} \rightarrow p(m-1) = (m-1) - 64\left[\frac{m-1}{64}\right] ,$$

$$r = 3\left[\frac{m-1}{64}\right] + j-1 \qquad \text{(B.5)}$$

where $\left[\frac{m-1}{64}\right]$ denotes fixed point division. Thus, $p(m)$ in the above equation is the remainder of $\frac{m}{64}$. This storage configuration in the PE's is achieved by loading U_{jm}, $m = 1,2, \ldots N$, on the disk in the sequential order $\bar{\bar{U}}_S$, $S = 1,2, \ldots 3N$ where

$$S = m + 128\,\frac{m-1}{64} + 64(j-1) , \qquad \text{(B.6)}$$

$$j = 1,2,3 .$$

The condition for $\bar{A}_{in,jk}$ to arrive in the processor containing $U_{jm_{n,k}}$ is expressed, using Eq. (B.5), as

$$\bar{A}_{in,jk} \rightarrow p\,(m_{n,k}-1) \qquad \text{(B.7)}$$

The PE row number r to be occupied by the various non-zero terms of the sparse matrix is somewhat more difficult to express because of the arbitrariness of the node numbering scheme. The node numbers n should increase monotonically (but not necessarily sequentially) with increasing row number r in each PE so that the row number n of the sparse matrix can be processed in ascending order. However, in general, no more than 27 of the 64 PE's will contain a $U_{jm_{n,k}}$ for which $\bar{A}_{in,jk}$ is non-zero for any particular matrix row number n. That is, only about one-third of the PE's will contain nodal

displacements that are adjacent to node number n in the spatially zoned continuum. Furthermore, a single PE may, in some instances, contain more than one neighbor nodal displacement but rarely more than nine.

When performing the multiplications that contribute to matrix row number n, there is no need to make the noncontributing PE's inoperative. Each noncontributing PE can simultaneously perform multiplications for the next higher matrix row number for which the PE will have a contribution. If the compressed matrix $[\underset{\approx}{A}]$ is loaded into the PE's in any proper sequence, this work-ahead scheme can be carried out by performing multiplications in each PE in the sequence that the $[\underset{\approx}{A}]$ terms are loaded from mass storage. This desired storage configuration in the various PE's is achieved by loading $\bar{A}_{in,jk}$; $n = 1,2, \ldots N$; $k = 1,2 \ldots \approx 27$, on the disk in the sequential order $\bar{\bar{A}}_S$, $S = 1,2, \ldots \approx 10 \times 27 \times N$ where

$$S = 64r + (p+1) + 64(3i+j-4) , \qquad (B.8)$$

$$i \text{ and } j = 1,2,3 ,$$

in which $r = 0, 1,2 \ldots \approx 10 \times 27 \times N/64$ and $p = 0, 1,2 \ldots 63$ are the row and PE number, respectively. The PE number is $p(m_{n,k}-1)$. The row number for each PE is expressed by summing all previous entries in the particular PE, i.e., the row number for PEp is expressed in terms of n and k by

$$r = 10 \sum_{n'=1}^{n-1} \sum_{\beta'=1}^{\approx 27} \delta_{pp'} + 10 \sum_{k'=1}^{k-1} \delta_{pp'} \qquad (B.9)$$

where $\delta_{pp'} = 0$ for $p \neq p'$ and $\delta_{pp'} = 1$ for $p = p'$ and where, just as in Eq. (B.7)

$$p' = p\left(m_{n',k'^{-1}}\right) . \qquad (B.10)$$

Every 10th term in the array $\bar{\bar{A}}_S$ starting with S = 1 is used to store two index numbers: $m = m_{n,k}$ to identify the nodal displacement that is to be multiplied and n to identify the matrix row to which the multiplication contributes, Eq. (B.3).

Using the storage schemes defined above, i.e., starting from a point in the computations in which $\{\underset{\sim}{U}\}$ and $[\underset{\sim}{A}]$ appear in their prescribed sequences on the disk, the sparse matrix multiplication of Eq. (B.3) proceeds as follows:

1. $\{\underset{\sim}{U}\}$ is loaded into core by PE rows using a single access.

2. The serially arranged version of $[\underset{\sim}{A}]$, defined Eqs. (B.7) – (B.10), is accessed and its loading is initiated. The terms flow into core by PE rows starting with row 0. After 30 rows are filled the loading is continued, uninterrupted, back at row 0 overwriting the previously loaded terms. The computations and manipulations of the following steps are carreid out before the matrix terms are overwritten.

3. Initialize $r_0 = -10$; $r_2 = 0$; $B_r = 0$ for r = 0, 1,2, ... 99; $n_0 = 1$.

4. Three sets of three multiplications and product summation are performed simultaneously in all processors.

$$B_{r_2+i} = B_{r_2+1} + \sum_{j=1}^{3} \bar{\bar{A}}_{r_{0}+j+3i-3} \bar{\bar{U}}_{r_{1}+j-1} \; ; \; i = 1,2,3$$

where

$$r_0 = (r_0+10)\left(1-\delta_{20,r_0}\right)$$

$$r_1 = \left[\frac{m-1}{64}\right]$$

$$r_2 = \begin{cases} r_2, & \text{if } n = B_{r_2} \\ r_2+4, & \text{if } n > B_{r_2} \end{cases}$$

$$n = \overline{\overline{A}}_{r_0} \text{ (first 32 bits)}$$

$$m = \overline{\overline{A}}_{r_0} \text{ (second 32 bits)}$$

Also, store matrix row number of the product contribution

$$B_{r_2} = n$$

Set $n_0 = n_0 + 1$

5. Check for the completion of matrix row number n_0. If $n_0 = MIN(n)$ return to step (4); otherwise, i.e., $n_0 < MIN(n)$ continue on to step (6). MIN(n) is the minimum b_0 among all 64 PE's.

6. Perform a row sum on B_i; $i = 1,2,3$ among those PE's for which $b_0 - n_0$. With only PEp operative, shift the result to B_{r_1+i-1}, $i = 1,2,3$ where

$$p = p(n_0-1)$$

$$r_1 = \left\lfloor\frac{n_0-1}{64}\right\rfloor$$

Operating only those PE's for which $b_0 = n_0$ set $B_r = B_{r+4}$ for $r = 0, 1, \ldots 99$.

7. If the next ten PE rows of the $[\underset{\sim}{A}]$ matrix have been loaded, return to step (4); otherwise return to step (5).

The computations and manipulations of steps (4) – (7) are displayed in Table A.1.

TABLE A.1 COMPUTATIONS AND MANIPULATIONS OF STEPS (4) – (7)

U_{jm} $\quad i = 1,2,3; \quad n = 1,2,3 \ldots N$

$\bar{U}_{pr}$ $\quad p = 0,1,2, \ldots 63$

$$= (S-1) - 64\left[\frac{S-1}{64}\right] = p(m-1)$$

$$r = R_0, R_0+1, R_0+1, \ldots R_0+3\left[\frac{N+63}{64}\right]$$

$$= R_0 + \left[\frac{S-1}{64}\right] = R_0+3\left[\frac{m-1}{64}\right] + i-1$$

$\bar{\bar{U}}_S$ $\quad S = 1,2,3, \ldots \approx 3N$

$$= m+128\left[\frac{m-1}{64}\right] + 64(i-1)$$

$A_{in,jm}$ $\quad (n,m) = 1,2,3, \ldots N; \; i = 1,2,3; \; j = 1,2,3;$

$\bar{A}_{in,jk}$ $\quad n = 1,2,3, \ldots N;$

$k = 1,2,3, \ldots 27$

$m = m_{n,k}$ data

$\bar{\bar{A}}_{pr}$ $\quad r = R_1, R_1+1, R_1+2, \ldots \approx R_1 + 10 \times 27 \times \frac{N}{64}$

$$= R_1 + 10 \sum_{n'=1}^{n=1} \sum_{k'=1}^{\approx 27} \delta_{pp'} + 10 \sum_{k'=1}^{k-1} \delta_{pp'}$$

$p = 0,1,2 \ldots 63$

$= p(m_{n,k}-1)$

$p' = p(m_{n',k'}-1)$

$\bar{\bar{\bar{A}}}_S$ $\quad S = 1,2,3, \ldots \approx$

$$= 64r + (p+1) + 64(3i+j-4)$$